LA PRIMERA REVISTA DE PROGRAMACION EN CASTELLANO

SOLO PROGRAMADORES

AÑO V, NÚMERO 52

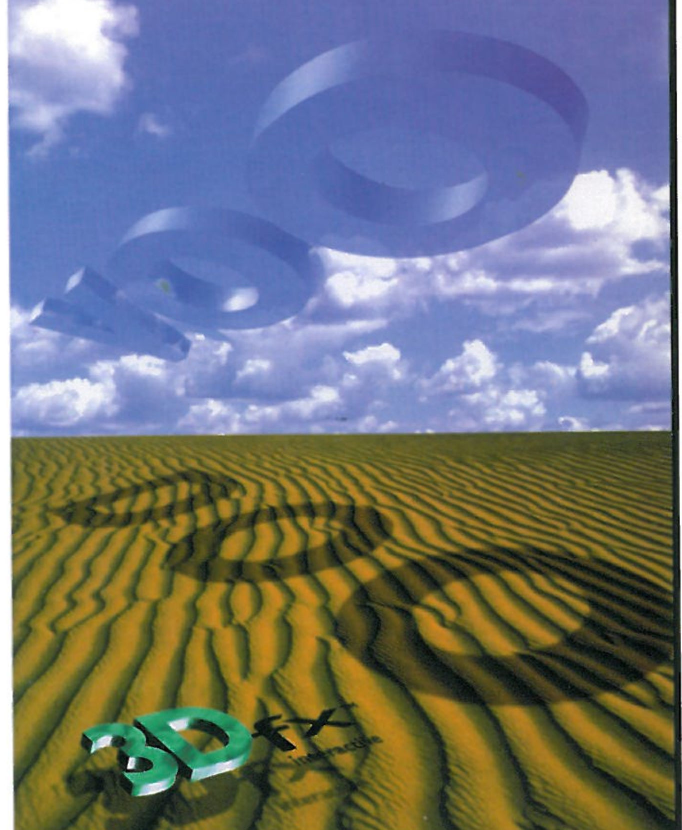
UNA PUBLICACIÓN DE:



975 Ptas.

PROGRAMACION GRAFICA

Sacar todo el rendimiento a una Voodoo 2 con las librerías 3D Glide



CONTENIDO DEL CD-ROM

- VELAZQUEZ VISUAL 3.0
- JAVA SERVLET DEVELOPMENT KIT 2.0
- TIS FIREWALL TOOLKIT
- SDK DE GLIDE

DESARROLLO CORBA
Marcando la diferencia entre la norma CORBA y DCOM

HTML DINAMICO
El efecto Scrolling

INTERNET
Cómo crear un buscador

JAVA
Construcción de aplicaciones gráficas multiplataforma con AWT

SEGURIDAD
Un cortafuegos con FireWall Toolkit

MULTIMEDIA
Reconocimiento de voz: DictaPad



SÓLO PROGRAMADORES

Número 52
SÓLO PROGRAMADORES
es una publicación de
TOWER COMMUNICATIONS
Director Editor
Antonio M. Ferrer Abelló
aferrer@towercom.es
Subdirector
Oscar Rodríguez Fernández
oscarrf@towercom.es
Coordinador Técnico
Eduardo De Riquer Frutos
eriquer@towercom.es
Coordinadora de Redacción
Erika de la Riva Arnáiz
eriva@towercom.es
Colaboradores
Constantino Sánchez, Juan J. Taboada,
José J. Mora, Javier Sanz, Adolfo Aladro,
Enrique de la Lastra, Enrique Díaz,
J.L. Alvarez.
Maquetación
Ana Isabel Madero Bocos
Tratamiento de Imagen
Clara Francés

Publicidad
Olga de Oteyza (Madrid)
Tel.: (91) 661 42 11
Pepín Gallardo (Barcelona)
Tel.: (93) 213 42 29

Suscripciones
Alicia Zazo
Tel. (91) 661 42 11 Fax: (91) 661 43 86
suscrip@towercom.es

Laboratorio
Javier Amado (Jefe)
jamado@towercom.es
Servicio Técnico
Manuel Hernando
mhernando@towercom.es

Preimpresión
Indes Color
Impresión
Gráficas Muriel
Distribución
SGEL
Distribución en Argentina / Chile / Colombia / México / Venezuela
Capital: Huesca y Sanabria
Interior: D.G.P.

TOWER COMMUNICATIONS
Director General
Antonio M. Ferrer Abelló
Director Financiero
Francisco García Díaz de Liaño
Director de Producción
Carlos Peropadre
Directora Comercial
Carmina Ferrer
carmina@towercom.es

Redacción, Publicidad y Administración
C/ Aragoneses, 7
28108 Pol. Ind. Alcobendas (MADRID)
Telf.: (91) 661 42 11 / Fax: (91) 661 43 86

Depósito legal: M-26827-1994
ISSN: 1134-4792
PRINTED IN SPAIN
COPYRIGHT 30-3-99



Una revista de tomo y lomo

Aunque a lo largo de la andadura de Sólo Programadores se han producido bastantes cambios, más o menos relevantes, con mayor o menor acierto, en esta ocasión no podía dejar de comentarles algunos detalles relacionados con el tema. La gran acogida del suplemento adicional dedicado al sistema operativo Linux ya provocó cambios importantes en esta revista, y en el momento actual, se avecinan otros nuevos. Estos cambios se deben al incremento de páginas de dicho suplemento y a su lanzamiento con un formato independiente al margen de Sólo Programadores, pero con una relación muy estrecha.

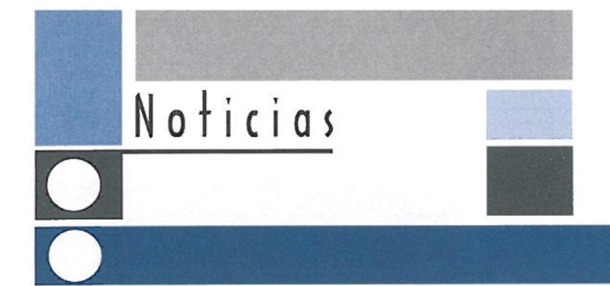
Como ya les indiqué anteriormente, el motivo principal de tan "dolorosa" despedida se debe fundamentalmente a la demanda suscitada por Sólo Linux, a la petición reiterada de un aumento progresivo de su extensión, así como la profundidad de sus artículos. De esta forma y dado que el objetivo inicial de este proyecto consistía en poner al alcance de cualquier usuario el sistema operativo con mejores perspectivas de futuro, la posibilidad de conseguir este objetivo y ofrecer temas avanzados chocaba con la falta evidente de espacio.

Llegados a este punto, la única alternativa razonable y satisfactoria para nuestros lectores consistía en iniciar dos caminos diferentes, pero a la vez paralelos. Y digo a todos los lectores, porque la otra alternativa se basaba en el mantenimiento conjunto de las dos revistas, a cambio de un incremento en el precio de portada. Aún tratándose de una posibilidad real y factible, se oponía a los lectores fieles a Sólo Programadores con un mínimo de interés por Linux, ya que tendrían que pagar más por acceder a la misma parcela de información útil para ellos.

En definitiva, esta nueva trayectoria intenta acercar nuestras revistas al mayor número de lectores, ampliando el abanico de posibilidades y sobre todo, concretando los temas para que todos dispongan de la alternativa más apropiada. La primera consecuencia, que a buen seguro responderá a las demandas de nuestro público, consiste en que se vuelve al formato inicial de la revista, con lomo y numeración visible. Sentimos mucho la variación sufrida y los trastornos ocasionados, pero muchas veces las alternativas pasan por encontrar una solución lo mejor posible para todos, aunque no se trate de la solución óptima. En cualquier caso, intentaremos que el valor añadido que supuso Sólo Linux a nuestra revista no se pierda con la separación, así que este mes les ofrecemos un segundo CD-ROM dedicado al interesante Visual Studio 6.0 y Visual J ++ 6.0.

SÓLO PROGRAMADORES





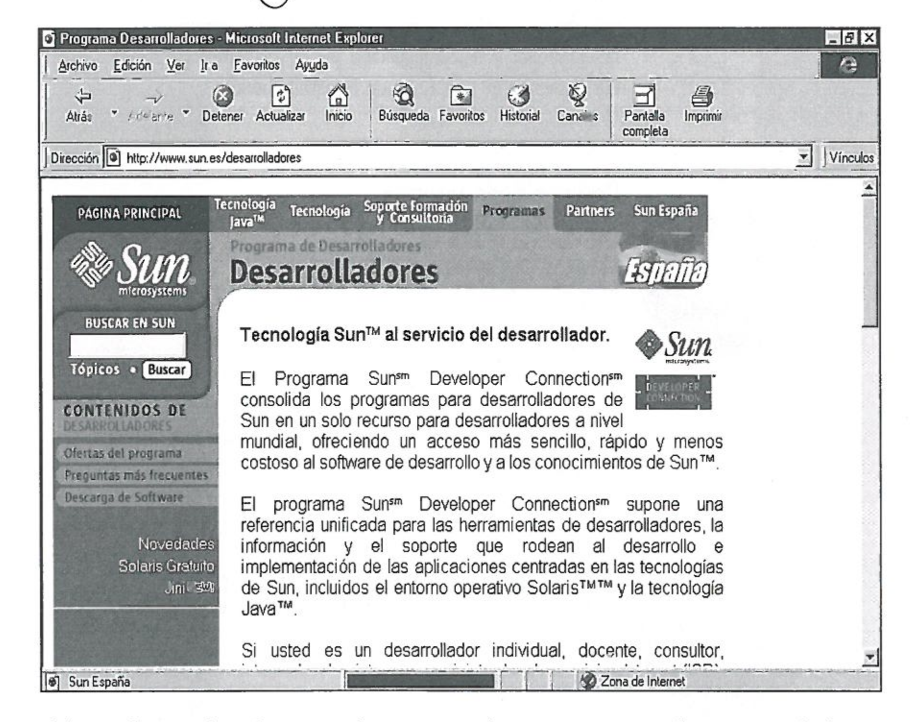
Sun unifica, simplifica y expande sus programas para desarrolladores así como su soporte

PRESENTACIÓN DE SUN DEVELOPER CONNECTION

Ya está disponible, según anunció la compañía Sun Microsystems, el programa Sun Developer Connection en nuestro país.

Este programa es una amplia iniciativa de información y recursos, dirigida a cubrir las necesidades de los desarrolladores de Sun en todo el mundo. Con él obtendrán de una manera muy sencilla los recursos e información que la compañía dispone para ellos.

Tanto los desarrolladores independientes como empresas, podrán acceder a los programas y ofertas de la compañía a través de un único punto de entrada en el sitio web o mediante un teléfono de asistencia gratuito (900 98 44 64).



Los desarrolladores que trabajan con tecnología Java o el sistema operativo Solaris, o ambos también pueden utilizar este punto de entrada, donde encontrarán información y recursos sobre cada una de las divisiones de la empresa Sun.

El programa ha sido diseñado para dar servicio a un gran número de desarrolladores y para escuchar su voz dentro del seno de la compañía. Incorpora más recursos para desarrolladores individuales.

Sun ofrece además un nuevo programa denominado Sun Info Essentials mediante el cual, se puede acceder a la información de productos y tecnologías , consejos y técnicas, boletines informativos etc...

También podrán contar con un servicio de suscripción llamado Sun Developer Essentials que proporciona a precio económico software, para aquellos que efectúan evaluación y desarrollo de programas comerciales.

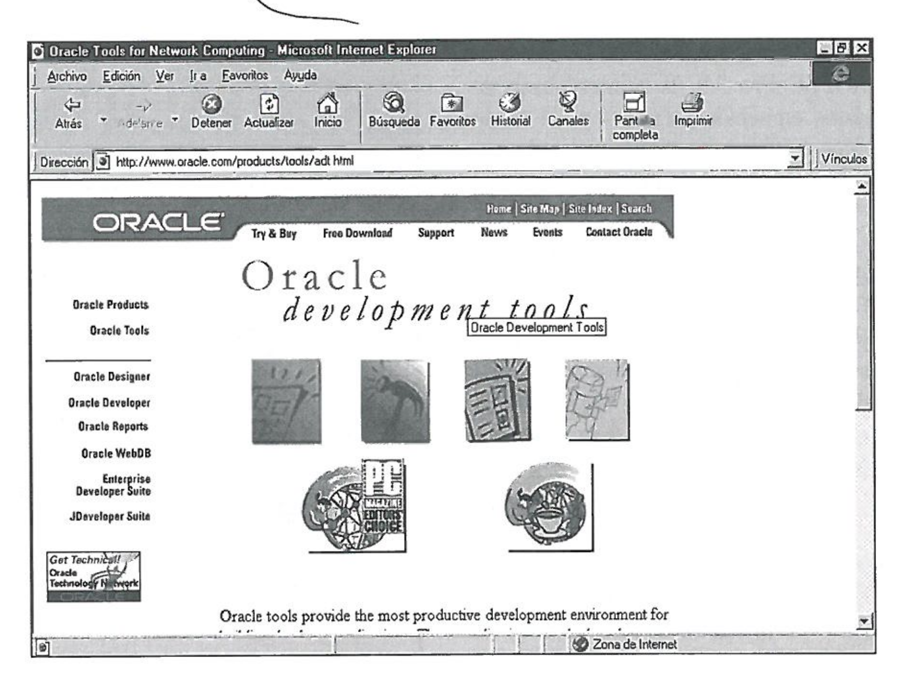
Asimismo diponen de una oferta de soporte técnico flexible, Sun Support Acces, con un amplio espectro de opciones de soporte que van desde una biblioteca online gratuita hasta contratos de soporte de ingenieros de Sun.

Todos aquellos desarrolladores que lo deseen podrán ponerse en contacto con la compañía en una dirección de correo electrónico, llamada la voz del desarrollador (eurodev@sun.com) a la que poder enviar consultas y sugerencias.

El sitio web de esta empresa, al que hemos llamado punto de encuentro, y desde el que se puede acceder para informarse sobre este programa dedicado a desarrolladores es www.sun.es/desarrolladores

ORACLE LANZA UNA HERRAMIENTA PARA MIGRAR DESDE LA BASE DE DATOS DE MICROSOFT A ORACLE 8

Oracle ha presentado una nueva herramienta, Oracle Migration Workbench, que permite la conversión instantánea de información almacenada en cualquier versión de SQL Server a la base de datos Oracle8 de Oracle. Esta nueva herramienta se distribuirá a finales de año sin ningún coste.



El proceso de migración desde la base de datos de Microsoft a la de Oracle es mucho más simple y rápido que el que se realiza entre anteriores versiones de SQL Server a SQL Server 7.0.

Oracle dispone ya de una versión beta de un conjunto de herramientas que amplía funcionalidades de Oracle8 con asistentes gráficos, cuyo objetivo es acelerar y hacer mucho más fácil y rápido el proceso de desarrollo de aplicaciones sobre Windows NT, ya que esto automatiza muchos de los procesos a realizar. Esta herramienta va a facilitar enormemente la migración de datos a Oracle8, tanto en plataformas NT como en otras muchas, sin que esto conlleve ningún tipo de problema.

La dirección de la página web de Oracle en la que puede ampliar toda la información que necesite conocer acerca de esta nueva herramienta es www.oracle.es

SUN INCORPORA LAS FUNCIONALIDADES DE LA NUEVA TECNOLOGÍA JAVA AL SOFTWARE SOLSTICE ENTERPRISE MANAGER

Sun Microsystems acaba de anunciar Solstice Enterprise Manager 3.0, una versión compatible con la tecnología Java de su software de gestión de redes distribuidas, multiprotocolo y con un alto grado de escalabilidad.

Esta versión aprovecha el entorno Java para conseguir más rapidez para la comercialización de nuevos servicios empresariales, manteniendo una escalabilidad ilimitada.

La nueva integración de Java Dynamic Management Kit en Solstice Enterprise Manager 3.0 permite gestionar las tecnologías actuales y la utilización de Java bajo una base de gestión común y coherente, además de facilitar el desarrollo rápido de aplicaciones.

Con las novedosas funcionalidades y mejoras de Solstice Enterprise Manager se podrán gestionar redes fiables, disponibles y escalables de millones de objetos,al mismo tiempo que se implementan nuevos productos y servicios de una forma más dinámica. Puede obtener mucha más información consultando la website www.sun.es

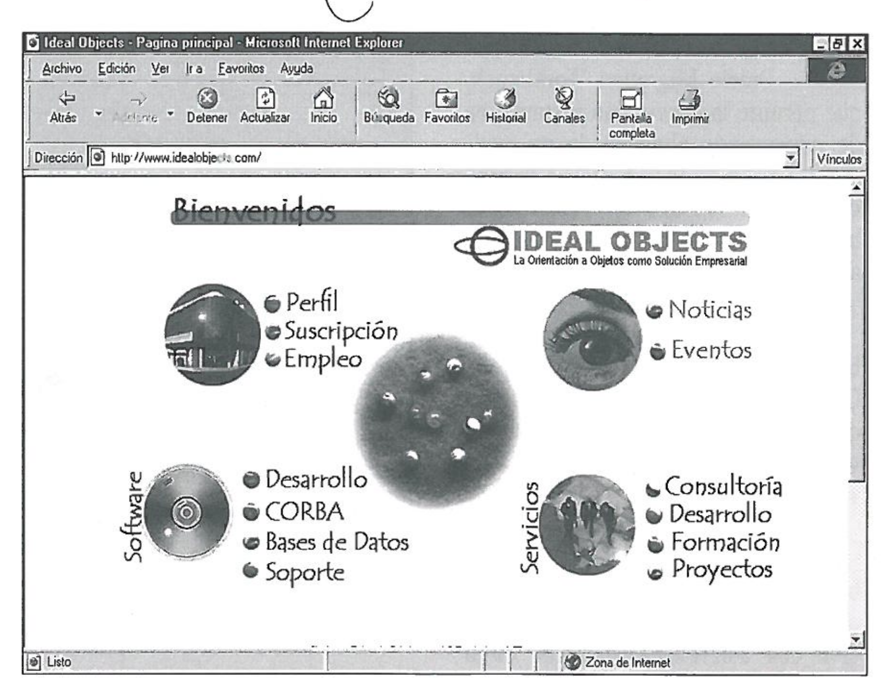
IDEAL OBJECTS PRESENTA LA PRIMERA BASE DE DATOS DE OBJECT DESIGN DISPONIBLE PARA APLICACIONES JAVA Y C++

Ideal Objects ha presentado ObjectStore PSE Pro 3.0, la primera base de datos embebida para Java y C++, de Object Design, proveedor líder de sistemas de almacenamiento.

Esta base de datos comprende dos pequeños sistemas de gestión de base de datos, además incluye un conjunto de herramientas de desarrollo visual, permitiendo a los desarrolladores hacer que sus aplicaciones embebidas funcionen más rápida y fácilmente.

ObjectsStore PSE Pro 3.0 es la primera base de datos embebida certificada tanto para Windows CE como para entornos Java.

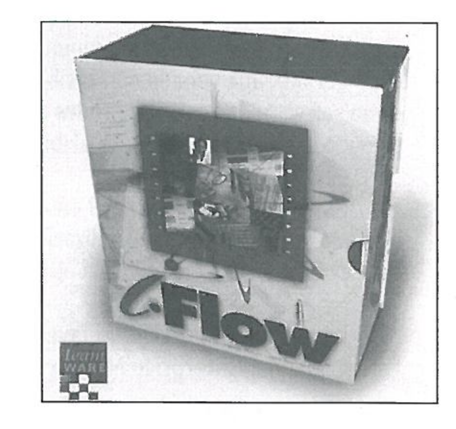
Igualmente se pueden gestionar datos de Java y C++ en sus formatos nativos, sin necesidad de mapeo adicional o código de traducción, lo que puede conseguir reducir el footprint total de la aplicación al 50% o incluso algo más. Puede consultar en la siguiente dirección web para ampliar la información sobre este producto www.idealobjects.com



TEAMWARE AMPLÍA SU OFERTA DE SOLUCIONES DE PROCESOS DE NEGOCIO CON I-FLOW, SU NUEVA PLATAFORMA DE DESARROLLO WORKFLOW BASADA EN JAVA

TeamWARE anuncia i-flow, una nueva herramienta de automatización de procesos de trabajo diseñada específicamente para su utilización a través de la Web.

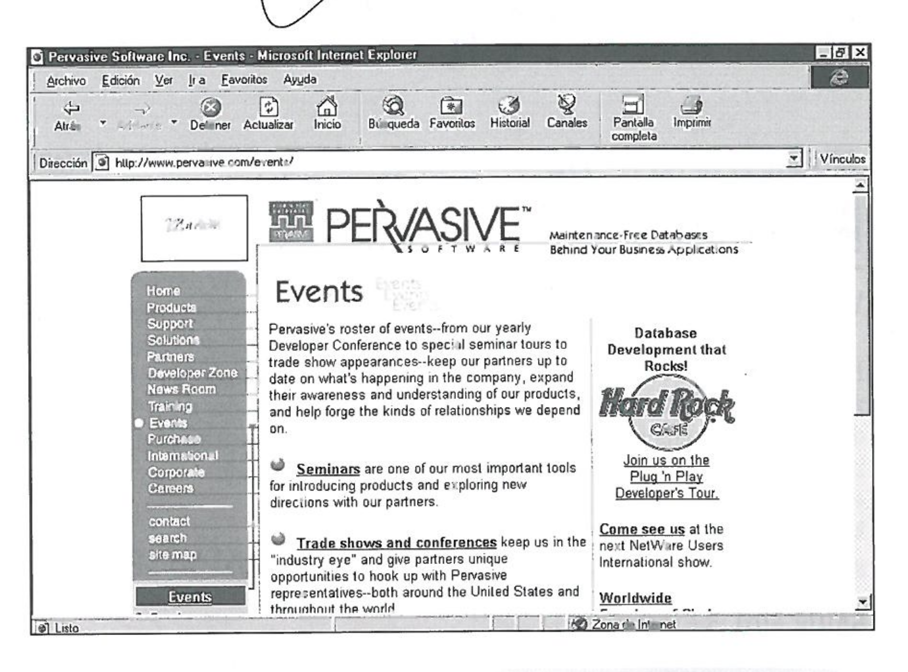
Su arquitectura permite una sencilla integración con todas las tecnologías de la Información tanto existentes como emergentes a través de Adapter Objects, que está basado en Java. i-Flow está actualmente disponible para el mercado norteamericano. La comercialización del producto en Europa comenzará a partir de Enero de 1999 , y estará disponible en castellano durante el primer trimestre del próximo año. Todos los datos sobre esta oferta están disponibles en la website www.teamware.com



PERVASIVE SOFTWARE ANUNCIA UNA GIRA MUNDIAL DE SEMINARIOS

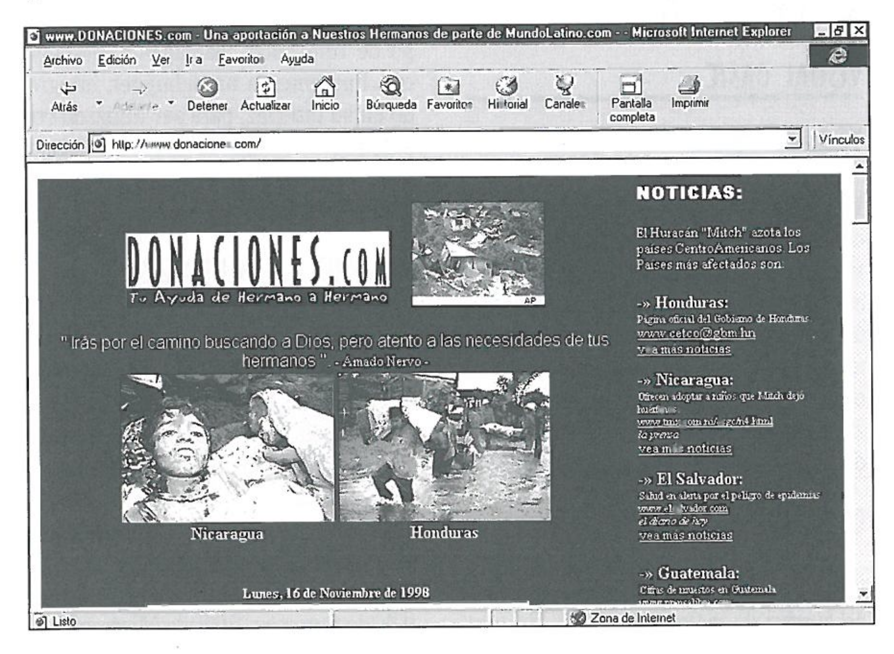
Pervasive desvela los detalles de su Gira Mundial para Desarrolladores "Freedom of Choice".

Estos eventos están previstos en más de 50 ciudades en mercados internacionales claves, ofreciendo a los desarrolladores de aplicaciones en el mundo una presentación en profundidad y gratuita del nuevo Pervasive. SQL Software Developers Kit, así como otros productos de Pervasive como Data Conversion Tool y una suite de desarrollo de aplicaciones para la web. Esta Gira se celebrará en lugares de ocio entre los que incluye a los conocidos Hard Rock Cafés. Más detalles e información visitando la página web www.pervasive.com/events



SOLIDARIDAD CON CENTROAMÉRICA

En estos momentos y con el objeto de dar a conocer, con detalles, las diversas organizaciones y entidades que están solicitando donaciones para los diferentes pueblos afectados por el huracán Mitch, MundoLatino.com y Web Enterprises han creado www.Donaciones.com



DESARROLLO EN GRUPO

IBM, Unisys y Oracle junto a otros suministradores de software y usuarios finales, han presentado una propuesta para la creación de un estándar que mejore el desarrollo de aplicaciones en grupo.

Esta especificación, denominada XML, está destinada a que los equipos de desarrolladores que trabajan con tecnología orientada a objetos y utilizan diferentes herramientas puedan enviar datos de programación a través de Internet. Esta nueva especificación ha sido enviada al Object Management Group (OMG), organismo que aprueba estándares de tecnologías orientadas a objeto.

HERRAMIENTAS DE PROGRAMACIÓN

ENTORNOS DE DESARROLLO

VELÁZQUEZ VISUAL 3.0

Herramienta de desarrollo rápido de aplicaciones de gestión y bases de datos. Una herramienta de desarrollo revolucionaria y puede ser una gran eleccción para aquellos que nunca hayan programado aplicaciones de gestión o de bases de datos.

GATOR EDIT 32 2.0

Potentísimo editor específico para programadores. Incluye una potente herramienta de búsqueda y sustitución, un diccionario corrector, capacidades de creación de marcas y puede mantener abiertos más de 100 ficheros a la vez. Dispone de un Portapales propio que aumenta la potencia de acciones de reutilización de código gracias al Copiar y Pegar.

JAVA SERVLET DEVELOPMENT KIT (JSDK) 2.0

Entorno para la creación de applets específicos para servidores Web. Potente solución de desarrollo de Sun que implementa todas las características para ser el estándar en programación de servlets. Este kit contiene un motor para la creación y testing de servlets, los códigos javax.servlet y documentación sobre la API.

PERL BUILDER 1.0A.6

El primer IDE para la programación en Perl bajo Windows. Perl Builder es una solución completa para la creación, compilado y testeo de scripts Perl. Además incluye un módulos que facilitan también la creación de scripts CGI que pueden ser usados para desarrollar programas Perl para otras plataformas. Con un desarrollo visual y de asistentes se pueden crear aplicaciones en Perl de manera sencilla. Ofrece una integración total entre Editor y Debugger.

SETUP SPECIALIST 98 SERVICE RELEASE 1

Desarrollo de procesos de instalación para aplicaciones 16 y 32 bits. La ultimísima versión de un sistema rápido y muy profesional de desarrollar instalaciones personalizadas para nuestros programas.

LENGUAJES

VISUAL BASIC

VBASSIST 5.02

15 potentes herramientas para añadir al entorno de desarrollo de Visual Basic. Entre otras se pueden encontrar un asistente para la creación de atajos de teclado, un portapapeles con histórico, un asistente para la alineación de componentes, un visor de proyectos, un asistente de recursos, un zoom mejorado, un asistente de propiedades, etc.

GCJRWINDIALOGS 1.0

Librería que permite mostrar 40 tipos de cuadro de diálogo distintos. Es una interesante utilidad que permite mostrar diferentes opciones del Panel de Control y del menú Inicio de Windows a través de varios tipos de cuadro de diálogo. Entre ellas reiniciar, ejecutar, ejecutar salvapantallas, propiedades, abrir con, etc.

ROBOPRINT 1.0

Control que permite imprimir y acceder a opciones de visualización. RoboPrint es un control de 32 bits que permite añadir la interactuación sobre las opciones de impresión y vista previa desde un programa Visual Basic.

JAVA

JIG 1.1

Entorno para la creación de aplicaciones Java. Se trata de un entorno que permite escribir, gestionar y compilar applets o programas escritos en Java. Incluye distintas vistas de clases, código de referencia, llamadas a métodos, etc. Implementa un debugger, integrado en su interfaz, para ser utilizado con el JDK 1.1.6. Dispone de las instrucciones de instalación para Windows 95/98/NT y UNIX.

JPADPRO 3.6 BUILD 226

IDE para la creación de applets Java y documentos HTML. Permite tanto el desarrollo en Java como para la edición de documentos escritos en HTML. Ofrece un método rápido para escribir, compilar, ejecutar y testear código de programación. Gracias a sus herramientas es muy sencillo encontrar bugs, localizar clases, permutar entre trabajar con los JDK de Sun y los SDK de Microsoft.

FREEJAVA 1.0

El IDE perfecto para el JDK de Sun. Es el entorno adecuado para enmascarar con un interfaz de usuario al JDK de Sun. Se trata de una utilidad que precisa de mínimos requerimientos y que ofrece una potencia más que interesante.

HTML

ACEEXPERT 2.54

Editor HTML especialmente indicado para desarrolladores expertos. Especialmente indicado para el desarrollo de páginas HTML que precisen de la potencia de applets Java y JavaScripts. Es la solución actual para los programadores que deseen incluir en sus páginas las últimas tecnologías.

BUTTONZ & TILEZ 1.5

Pontentísima utilidad para el diseño de botones y fondos Web. Bajo un interfaz gráfico muy intuitivo permite realizar elementos de diseño Web (botones, fondos, líneas, etc) de gran calidad. Es sencillo implementar un diseño profesional que dará modernidad a nuestros desarrollos HTML.

HTML VALIDATOR 3.01

Utilidad de gran calidad para la validación de código HTML. El programa chequea el código en busca de errores sintácticos y genera un listado con todos los problemas encontrados. Dispone a su vez de otras herramientas que permiten ordenar los tags HTML, crear plantillas para los documentos, convertir los textos para que sean reconocidos en distintas plataformas, etc.

OTROS

MICROSOFT AGENT ACTIVEX CONTROL 2.0

Permite añadir personajes interactivos a los desarrollos Web. Es un kit que permite la integración de personajes animados dentro de desarrollos HTML o Windows. Permite al desarrollador crear asistentes que realicen acciones de ayuda o formación para incorporar una nueva forma de interacción. Agent incluye soporte de introducción de órdenes a través del teclado, el ratón e incluso a través de voz.

ACTIVEX MANAGER 1.2

Permite la gestión de los controles ActiveX instalados en el sistema. Interesantísima utilidad que ofrece la posibilidad, tanto a usuarios como a desarrolladores, de poder gestionar los controles ActiveX instalados en el sistema.

HACKMAN 3.0

Editor multifuncional con capacidades de disassembler. El contenido de cualquier fichero puede ser visualizado y modificado en formato hexadecimal, ASCII, binario, octal y decimal. Ofrece opciones de buscar y reemplazar, comparación, encriptación, etc.

HERRAMIENTAS Y UTILIDADES

BLACK HOLE ORGANIZER 1.1

Potente sistema de base de datos para la organización de la información personal. Posibilita la creación de bases de datos personalizadas que pueden contener información como notas, recordatorios y todo tipo de documentos de texto.

CD COPY 4.5

Convierte pistas de un CD de música en ficheros de sonido. Permite guardar los ficheros en los formatos: AU, RAW, WAV, Yamaha VQF, RealAudio, MPG, MPA (MPEG 2 Layer 3) y MP3 (MPEG 1 Layer 3).

EXAMDIFF 1.6

Utilidad para la comparación de dos ficheros con código. Permite guardar los ficheros en los formatos: AU, RAW, WAV, Yamaha VQF, RealAudio, MPG, MPA (MPEG 2 Layer 3) y MP3 (MPEG 1 Layer 3).

FILE INVESTIGATOR 1.22

Permite reconocer los formatos de más de 300 tipos de ficheros. Utilidad muy interesante que puede ayudar al desarrollador a indentificar tipos de ficheros con distintas extensiones.

REDES

LOCALES

DNTOOLS V1.2

Colección de herramientas para mejorar el Acceso Telefónico a Redes. Optimiza las propiedades para mejorar el rendimiento de Internet y redes locales.

CHEYENNE FAXSERVE FOR WINDOWS NT

Programa servidor de envío y mantenimiento de documentos fax. Permite el envío o recepción de un fax a cualquiera de los componentes de una red local. Para el envío bastará con que realice la misma operación que cuando imprime un documento. Posibilita la operación de mandar fax desde cualquier aplicación que permita imprimir.

DISTRIBUIDAS

COM EXPLORER 1.5

Administración en el servidor de ficheros .EXE, .DLL y ActiveX. Con un interfaz similar al del Explorador de Windows ofrece a los desarrolladores una forma sencilla de gestionar las comunicaciones registradas en el servidor.

INTERCOM 2.0

Kit para potenciar los conocimientos sobre protocolos de comunicaciones. Intercom usa PPP y TCP/IP para conectar al servidor Internet y ejecutar una simple aplicación enviando y recibiendo paquetes de datos por correo.

CHEYENNE CD SERVER 1.0

Integración de jukeboxes de CD-ROM con NT. Permite la utilización de torres de lectores de CD-ROM para ser usadas en estaciones de trabajo, servidores compartidos o redes de área local bajo Windows NT.

DELPHI ANIMATED CURSOR

Cursor animado para NT. Curiosa aplicación desarrollada en Delphi. Es un componente freeware y se distribuye con el código fuente.

DOCUMENTACIÓN / TUTORIALES

C LANGUAGE TUTORIAL

Curso dividido en 13 capítulos sobre la programación de punteros en C. Cada uno de los capítulos está perfectamente documentado y apoyado por cuestionarios que refuerzan el aprendizaje.

MS DOS INTERRUPT LIST 59

Interesante listado de todas las funciones MS DOS disponibles a través de las interrupciones del sistema. La guía incluye más de 9200 entradas y 5000 tablas. Se incluyen mapas de memoria de CMOS y BIOS, I/O ports, I2C-bus devices, y modos protegidos de gestión del sistema. Entre ellas se encuentran funciones no documentadas ni por el propio Microsoft.

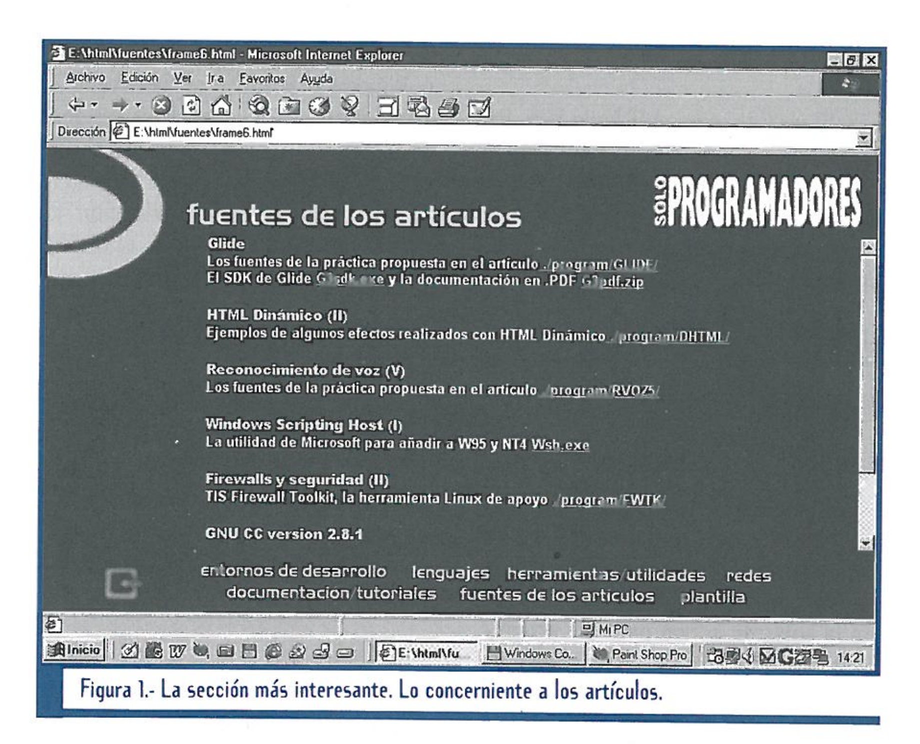


Figura 1.- La sección más interesante. Lo concerniente a los artículos.

VB 5.0 FLASH CARDS 2.0

Cuestionarios y fichas para dominar Visual Basic 5.0. Es ideal para aquellos que ya conocen Visual Basic pero que precisan de conocimientos superiores. Es un programa muy utilizado en EEUU por aquellos desarrolladores que se presentan a los exámenes para ser programador homologado por Microsoft.

VISUAL BASIC FOR KIDS 1.0

Tutorial interactivo que permite el aprendizaje de Visual Basic a niños mayores de 10 años. Se basa en un método de aprendizaje muy sencillo a base de ejemplos y procesos desarrollados paso a paso.

WORDCOMMAND 1.0

Utilidad especial para crear vocabularios específicos para documentación y tutoriales. Permite construir un vocabulario personalizado que puede servir como ayuda personal o para la documentacion de programas y proyectos. Por cada palabra es posible incluir una definición.

FUENTES DE LOS ARTÍCULOS

Glide:
Los fuentes de la práctica del artículo. El SDK de Glide G3sdk.exe y la documentación en .PDF G3pdf.zip

HTML Dinámico (II):
Ejemplos de algunos efectos realizados con HTML Dinámico.

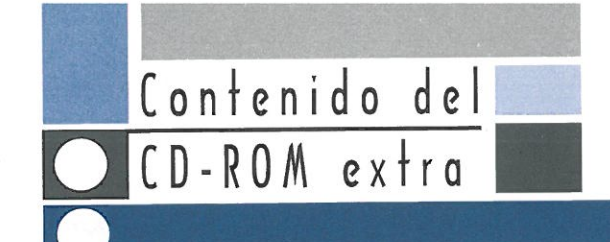
Reconocimiento de voz (V):
Los fuentes de la práctica propuesta en el artículo.

Windows Scripting Host (I):
La utilidad de Microsoft para añadir a W95 y NT4 Wsh.exe

Firewalls y seguridad (II)
TIS Firewall Toolkit, la herramienta Linux de apoyo.

GNU CC version 2.8.1
Unos de los compiladores de C más potentes.

ORBacus 3.1 para C++ y para Java
Hemos incluido ambas versiones.



CD- ROM ESPECIAL DE MS VISUAL STUDIO 6.0

Mucho esfuerzo nos ha costado pero al final podemos ofreceros un CD-ROM que hará las delicias, sobre todo, de los seguidores del lenguaje Java. Disfrútalo.

VISUAL STUDIO 6.0

Como no podía ser menos este mes os incluimos otro CD-ROM extra con la revista, esta vez dedicado a una herramienta que estamos seguros será de gran interés para vosotros.

Se trata de un CD específico para desarrolladores que incluye información técnica sobre la suite de programación Microsoft Visual Studio 6.0, una completa suite de herramientas que satisface las necesidades técnicas y de negocio del desarrollo empresarial. Esta suite la componen una serie de herramientas imprescindibles para todo profesional como son Visual Basic, Visual FoxPro, Visual InterDev y Visual J++.

El CD-ROM incluye toda la información técnica relativa a los programas que componen esta suite y además os entregamos una versión totalmente operativa durante treinta días de Visual J++ 6.0

MS VISUAL J++ 6.0

Como primicia de fin de año os ofrecemos una versión completamente operativa de Visual J++ 6.0. Durante treinta días podréis utilizar esta herramienta que incluye muchas y nuevas prestaciones diseñadas para ayudar a los desarrolladores a construir aplicaciones profesionales para Internet con Java y conseguir dominar el lenguaje del futuro.

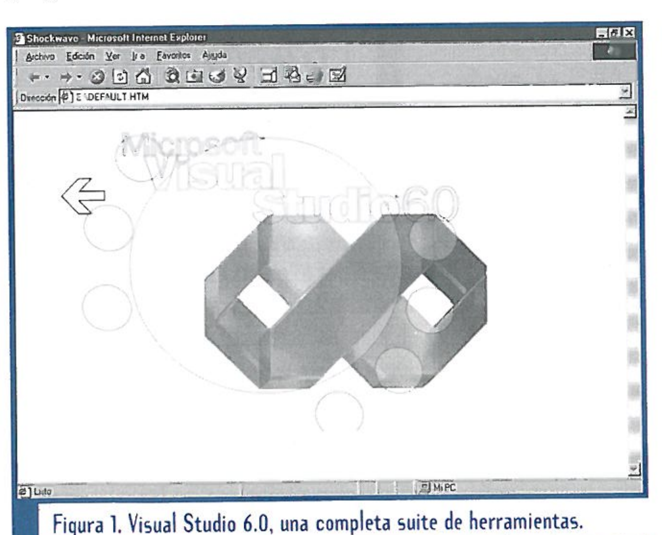


Figura 1. Visual Studio 6.0, una completa suite de herramientas.

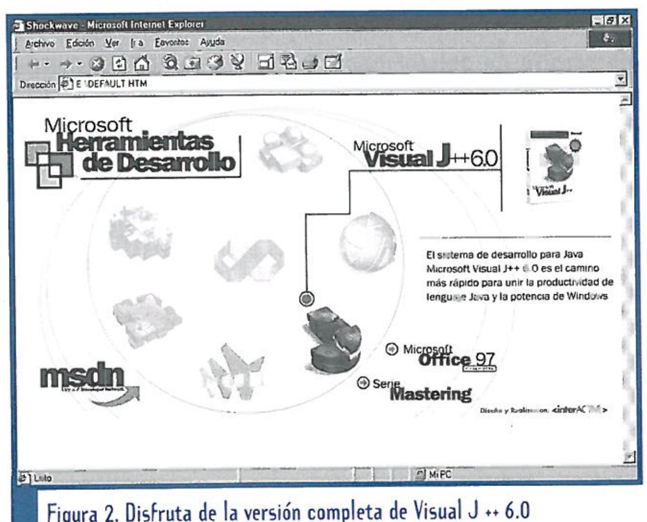


Figura 2. Disfruta de la versión completa de Visual J ++ 6.0

SÓLO PROGRAMADORES

Introducción al AWT

Javier Sanz Alamillo (jsanz@a01-unix.uc3m.es)

La utilización de AWT ha permitido a los desarrolladores Java construir aplicaciones gráficas multiplataforma que han animado a los grandes fabricantes de software a crear potentes entornos visuales, además de una gran variedad de applets que decoran miles de páginas web en Internet.

Gracias a las interfaces gráficas se terminaron los tiempos en que las aplicaciones requerían operaciones desde la línea de comandos. Por ello, todo lenguaje que se precie debe constar de una librería gráfica para el desarrollo de estas aplicaciones. En Java, esta librería se denomina AWT. Otra de las ventajas que ofrece esta librería consiste en la posibilidad de crear applets para las páginas web.

El AWT es una librería para el desarrollo de aplicaciones gráficas.

EL AWT

El AWT (Abstract Windowing Toolkit) es un conjunto de clases Java para la construcción de interfaces gráficas de usuario. Con el AWT se pueden crear ventanas, trabajar con imágenes, dibujar líneas, etc. Todo este conjunto de posibilidades permite que el programador no tenga que preocuparse por detalles tales como el control del movimiento del ratón o la visualización de componentes en pantalla. Todas estas clases Java que forman el AWT se encuentran agrupadas en el paquete java.awt.

ESTRUCTURA DEL AWT

Las estructura del AWT se puede dividir en dos grandes bloques:

- Una parte que gestiona la interfaz de usuario, como son los botones, etiquetas, barras de desplazamiento, ventanas, etc.
- Una parte gráfica que gestiona los procesos relacionados con las imágenes y los gráficos, tales como líneas, círculos, elipses, etc.

La interfaz de usuario a su vez se compone principalmente por contenedores en los que se agrupan componentes, que son los controles básicos (botones, etiquetas, etc. El AWT usa gestores de composición (layout managers) para situar los componentes en unos elementos que se denominan contenedores, además del control de su tamaño y posición. Los gestores de composición especifican qué tipo de visualización se obtendrá en función de los componentes insertados.

LOS COMPONENTES

Un componente es el objeto fundamental de la interfaz de usuario en Java. Todo lo que se puede visualizar en una aplicación es un componente del AWT, por lo que podemos incluir las ventanas, botones, barras, casillas, etc. Para poder utilizar dichos componentes se deben asociados a un contenedor. Un contenedor agrupa componentes, los sitúa para su visualización e incluso los asocia con un determinado dispositivo. Todos los componentes son clases derivadas de java.awt.Component.

La funcionalidad de los componentes puede establecerse en dos categorías: apariencia y comportamiento. Un componente contiene métodos y variables que controlan su apariencia. Esto incluye si el componente es visible o no, su tamaño y localización, etc.

La interfaz de usuario está compuesta por contenedores que agrupan componentes

El comportamiento de un componente se refiere a la forma en que reacciona frente a los eventos producidos por un usuario. Cuando un usuario realiza una determinada acción sobre cualquier componente, como pulsar una tecla del ratón, moverlo, etc., el AWT genera una evento asociado a dicho componente llamando a los métodos de gestión de eventos. Los eventos se suelen clasificar por el tipo de suceso (movimiento, pulsación, etc.)y entonces en función del componente y del evento se puede realizar una determinada acción. A continuación se describen los componentes básicos disponibles mediante el AWT.

BOTONES

Para crear un botón solo hay que crear un objeto de tipo Button con el literal que aparecerá encima del botón. Cuando se genera el botón, se obtiene el handle a ese objeto que permitirá gestionarlo posteriormente. La creación de un botón podría realizarse de la siguiente forma:

```
public class Boton extends Applet {
Button miBoton1 = new Button ("Boton1");
public void init () {
add ( miBoton1 );
        }
}
```

Para comprobar el resultado lo más apropiado en este caso consiste en crear una página HTML e incluir en ella el botón creado, pudiendo visualizarla mediante el appletviewer. Para poder mostrar los componentes utilizaremos como contenedor un applet por su sencillez de implementación.

CAMPO DE TEXTO

Un campo de texto, componente tipo Textfield, permite que el usuario teclee algún dato, que podremos asociar a una determinada variable. Un campo de texto se crea mediante la definición de un tamaño y un texto predefinido, que puede ser vacío. Veamos un ejemplo:

```
public class ctexto extends Applet {
        TextField mictexto = new TextField (
        "Texto inicial ...", LONG_CTEXTO);
        public void init () {
          add ( mictexto );
        }
}
```

ÁREA DE TEXTO

Un área de texto es un componente similar al campo de texto excepto que puede tener múltiples líneas y más funcionalidad, como por ejemplo que se puede insertar un dato en cualquiera de las filas del área, añadir filas, etc. Gracias a las mejoras en el AWT 1.1 se pueden construir áreas de texto en las que aparezca la barra de desplazamiento vertical, horizontal o ninguna. Mediante la siguiente sentencia se crea un componente de este tipo:

```
TextArea miatexto = new TextArea ( "",
        FILAS_ATEXTO, COLUM_ATEXTO );
```

ETIQUETAS

Las etiquetas, Label, sirven para visualizar un texto estático, que no será modificado, por lo que permite asociar un texto con un determinado componente. Una etiqueta tiene la alineación respecto a un componente en función del gestor de composición. Veamos un ejemplo con una etiqueta que aparece a la izquierda:

```
public class etiqueta extends Applet {
        Label mietiqueta = new Label("Este es
        un mensaje estatico");
         public void init () {
         setLayout ( new FlowLayout( FlowLa-
        yout.LEFT, 10,10 ));
         add ( mietiqueta );
        }
}
```

BOTONES DE MARCACIÓN

Un botón de marcación (Checkbox) se utiliza en situaciones en las que hay que elegir determinadas opciones y donde es posible elegir más de una. Cada opción creada requiere un literal asociado. El método principal de este tipo es getState(), que devuelve si el botón ha sido marcado o no. El siguiente ejemplo muestra su utilización:

```
public class bmarca extends Applet {
        Checkbox micheckbox1 = new Checkbox
        ("Rojo");
        Checkbox micheckbox2 = new Checkbox
        ("Verde");
        public void init () {
          add ( micheckbox1 );
          add ( micheckbox2 );
        }
}
```

Como se observa, los botones de marcación no son excluyentes entre sí.

BOTONES DE SELECCIÓN

Los botones de selección se pueden agrupar de forma que se crea un botón de radio (CheckboxGroup), que son agrupaciones de botones CheckBox en las que siempre hay un único botón activo. Su comportamiento es idéntico al de los botones de marcación, pero no se puede seleccionar más de

una opción a la vez. Pruebe con el siguiente ejemplo:

```
public class bradio extends Applet {
CheckboxGroup Radio;
        public void init () {
        Radio = new CheckboxGroup();
          add ( new Checkbox ("opc1",Radio,true
        ));
          add ( new Checkbox ("opc2",Radio,false
        ));
        }
}
```

Conviene indicar que existe el método *getSelectedCheckbox()* mediante el cual obtendremos la selección activada para un determinado grupo de opciones.

BOTÓN DE LISTA

Los botones de listas (Drop-down list), son componentes de tipo Choice que permiten elegir un único elemento de una lista de opciones. Veamos un ejemplo:

```
public class blista extends Applet {
        Choice botonlista ;
        public void init () {
          botonlista = new Choice ();
          botonlista.add ( "opcion1");
          botonlista.add ( "opcion2");
        }
}
```

Los botones de lista disponen del método getItem() que devuelve la cadena cuya posición de la lista le estamos indicando. La posición del elemento seleccionado se obtiene mediante el método getSelectedIndex(). Con la combinación de estos dos métodos se puede obtener el literal seleccionado de la lista.

LISTAS

Las listas son componentes muy diferentes al botón de lista y no sólo por su apariencia. Mientras que en un botón de lista se puede seleccionar una única opción, en una lista se pueden hacer múltiples selecciones. Veamos un ejemplo:

```
public class lista extends Applet {
        List milista ;
        public void init () {
          milista = new List(2, false );
          milista.add ("Opcion1");
          milista.add ("Opcion2");
          add ( milista );
        }
}
```

La creación de la lista mediante new List(2,false) indica que nuestra lista visualizará dos elementos y que no se permitirá la selección múltiple. En caso de que el usuario desee activar la selección múltiple, el método getSelectedItems() devuelve una tabla con todas las filas que se encuentren seleccionadas.

CANVAS

Los Canvas son simplemente áreas gráficas vacías donde se puede dibujar. Tienen un funcionamiento diferente al resto de los componentes, es decir, no tienen ninguna funcionalidad implícita y no atienden a los eventos, por lo que son relativamente poco usadas.

LOS CONTENEDORES

Como se mencionó anteriormente, para poder utilizar cualquier componente es necesario asociarlos a un contenedor. La misión de un contenedor consiste en agrupar distintos componentes y situarlos según un gestor de composición para su visualización. Existen diferentes contenedores tal y como se describen a continuación.

PANEL

El contenedor de tipo Panel es el más simple de todos. Tan sólo se encarga de ofrecer un lugar donde situar componentes, que pueden ser a la vez contenedores de tipo Panel. La forma de añadir un Panel en un applet es la siguiente:

```
Panel p = new Panel();
add ( p );
```

Aunque a priori puede carecer de funcionalidad es muy utilizado cuando es necesario separar gran variedad de componentes de un applet para su posterior manejo y actualización. Se pueden anidar contenedores tipo Panel de la siguiente forma:

```
Panel p = new Panel();
add (p);
Panel sp = new Panel();
p.add (sp);
```

WINDOW

La clase window es una clase abstracta que define una ventana en un applet o en una aplicación. Es un tipo de ventana clásica sin título ni borde. Al ser una clase abstracta no se pueden crear objetos, pero si sirve de base para otro contenedor muy útil, el contenedor Frame.

FRAME

El contenedor Frame es la implementación más simple de una ventana, ya que sólo define un borde y un título, aparte de que se pueden ir incorporando diferentes componentes. Además se le puede añadir una barra de menú, por lo que se trata de la clásica ventana que conoce todo usuario.

Para crear una ventana debemos derivar nuestra clase de la clase Frame e insertar los componentes que deseamos. Además se debe definir un gestor de componentes. Veamos un ejemplo:

```
public class ventana extends Frame {
        int tam_x = 200 ;
        int tam_y = 200 ;
public ventana ( String titulo ){
```

```
        super ( titulo );
        pack();
         resize ( tam_x, tam_y );
        show();
        }
        public static void main ( String args [] ) {
         new Ventana ( "Mi Titulo ");
        }
}
```

Al ejecutar el programa se obtienen obtenemos una ventana simple con un título de nuestra elección, que se impone mediante la llamada super() a la clase base.

Los métodos pack() y show() se necesitan para la correcta visualización de los componentes, ya que inicialmente, en un frame los componentes son invisibles. Por ello, se invoca a pack() que hace que los componentes se ajusten a unos tamaños adecuados en función del gestor de composición.

Como veremos más adelante el gestor de composición por defecto es BorderLayout.

Para crear una ventana debemos derivar nuestra clase de la clase Frame

Por otra parte, el método show() hace que los componentes sean visibles. Mediante la función resize() podemos definir el tamaño de presentación de nuestra ventana.

Una característica importante de los Frames consiste en la posibilidad de incluir barras de menú. La forma de añadir este componente a un contenedor es la siguiente:

```
MenuBar menu = new MenuBar("mimenu");
menu.add ("abrir");
menu.add ("cerrar");
ventana.add (menu);
```

VENTANA DE DIÁLOGO

Una ventana de diálogo es una ventana que surge de otra ventana, como la clásica ventana de "Acerca de...". El propósito de este tipo de contenedor es tener un nuevo espacio donde realizar una tarea sin interferir sobre la ventana origen. Suelen tener un borde y ser de tipo modal, es decir, cuando se visualiza no se puede cambiar a la ventana madre hasta no cerrar la ventana actual. A continuación se muestra el código necesario para crear una ventana de diálogo:

```
public class por extends Dialog {
        por ( Frame vent_ant )   {
         super ( vent_ant, "Por ... ", true );
         this.setResizable ( false );
         setLayout (new BorderLayout());
         setBackground ( Color.white);
         Panel p = new Panel();
         p.add ( new Button("Aceptar"));
         Panel p2 = new Panel ();
         p2.add ( new Label( " Calcula Cuadrados
        ... ", Label.CENTER));
         add ("North", new Label(""));
         add ( "South",p);
         add ("Center",p2);
         pack();
         resize ( 150,150);
         show();
        }
}
```

Como se ha comprobado el conjunto de contenedores es más que suficiente para abarcar el conjunto de necesidades visuales que se presentan hoy día.

GESTIÓN DE EVENTOS

Como se mencionó anteriormente, una de las propiedades de los componentes es que son capaces de responder a las acciones producidas por el usuario, ya sea el movimiento del ratón o pulsaciones del teclado. Estas acciones se denominan eventos, los cuales crean un objeto de tipo Event de forma automática que facilitará su manejo.
Un objeto de tipo Event consta de dos atributos: un identificador del tipo de evento y el componente sobre el que se produce. Gracias a esta construcción se puede saber qué tipo de evento se ha producido y sobre qué componente se ha realizado. Con la información anterior y con el método handleEvent() podemos atender los eventos que se produzcan, ya que este método se ejecuta en cuanto se produce un evento:

```
public boolean handleEvent ( Event evt )
```

Y sólo queda para el programador comprobar qué componente ha sido el afectado, ver que tipo de acción tiene asociada y realizar la tarea correspondiente. Por ejemplo, para detectar la pulsación de un botón se tendría el siguiente código:

```
public class controla extends Applet {
        Button miBoton = new Button("Boton");
        public void init () {
         add ( miBoton );
        }
        public boolean handleEvent ( Event evt )
        {
         if ( evt.target instanceof miBoton ) {
                // Boton pulsado
         }
        }
        return ( super.handleEvent(evt));
        // main ...
}
```

Es importante que siempre se finalice con una llamada:

```
super.handleEvent(evt)
```

Para asegurarse de que cualquier evento que no es controlado de forma explícita por el programador sea atendido en cualquier caso de forma genérica.

Conviene destacar que existen una serie de métodos específicos para ciertas acciones, entre los que sobresalen algunos como:

```
action ( Event evt, Object obj );
mouseDown (Event evt, int x, int y);
mouseEnter (Event evt, int x, int y);
keyUp ((Event evt, int key );
```

Para encontrar una lista más completa y detallada sólo hace falta consultar la documentación incluida en cualquier JDK.

LOS GESTORES DE COMPOSICIÓN

Los componentes son colocados por los gestores de composición (layout managers), los cuales deciden cómo se situarán los componentes que han sido incluidos en el mismo. El tamaño, forma y posición de cada uno de ellos será diferente en función del gestor que se utilice, debido a que los gestores de composición adaptan las proporciones de los componentes al contenedor al que se aplican, ya sea este un applet o la clásica ventana. Así, dado un contenedor cualquiera, el gestor de composición se determina mediante una llamada al método setLayout() que permite elegir entre los diferentes gestores. Veamos a continuación los principales gestores de composición.

El gestor FlowLayout organiza los componentes de forma que los enlaza centrándolos horizontalmente

FLOWLAYOUT

Este gestor es el más simple y organiza los componentes de forma que los enlaza centrando horizontalmente, de izquierda a derecha hasta que el espacio disponible se encuentre ocupado. A partir de ese momento continúa añadiendo por debajo los componentes anteriores, repitiendo el proceso de rellenado.

Es el gestor por defecto de los applets. Veamos un ejemplo:

```
public class Lflow extends Applet {
        public void init () {
          setLayout ( new FlowLayout () );
          for ( int i = 0 ; i < 6 ; i++)
                  add ( new Button ("Boton"+i));
        }
}
```

Mediante *setLayout(new FlowLayout())* podemos seleccionar el tipo de gestor. A continuación insertamos seis botones uno tras otro mediante el método add() que requiere como parámetro el componente, en este caso el botón.

BORDERLAYOUT

Este gestor utiliza una disposición algo más compleja que la anterior. Utiliza cinco zonas para colocar los componentes, que se determinan mediante las coordenadas norte, sur, este, oeste y centro. Así, la parte norte ocupa la parte superior del contenedor, la sur la inferior y así hasta la parte centro, que rellena el espacio restante. Este es el gestor utilizado por defectto para los contenedores de tipo Frame y Dialog. El siguiente ejemplo muestra la aplicación de este gestor:

```
public class Lborder extends Applet {
        public void init () {
          setLayout ( new BorderLayout () );
          add ("North", new Button("Norte"));
          add ("South", new Button("South"));
          add ("Center", new Button("Centro"));
        }
}
```

Como se observa, el método add() requiere primero la zona de posición y a continuación el componente que se desea insertar.

GRIDLAYOUT

Este gestor de composición es el que proporciona una mayor facilidad para situar los componentes. Gracias a él se crea una tabla con el número de filas y columnas que se desea tener de izquierda a derecha y de arriba a abajo.

```
public class Lgrid extends Applet {
        public void init () {
          setLayout ( new GridLayout (2,2) );
          for ( int i = 0 ; i < 4 ; i ++ )
          add ( new Button ("boton"+i));
          }
}
```

En este ejemplo que hemos visto observamos que GridLayout tiene como parámetros las filas y columnas que se van a utilizar.

OTROS LAYOUTS

Al igual que GridLayout, existe GridBagLayout, que se diferencia respecto al en que los componentes no necesitan tener el mismo tamaño. Se puede decir que se trata del más versátil de todos los gestores, pero también el menos utilizado por tener una mayor complejidad.

Se puede utilizar el gestor CardLayout cuando se tienen que colocar distintos componentes en la misma zona de un contenedor.

CONCLUSIONES

Gracias al AWT se pueden desarrollar aplicaciones visuales en Java, lo que cubre el vacío que existía inicialmente para este lenguaje. Consta de un gran número de componentes, cuyo sencillo manejo y utilización no requiere más que unas cuantas líneas de código, lo que anima a desarrollar nuevas aplicaciones visuales.

Reconocimiento de voz (V)

Constantino Sánchez Ballesteros (constantino@nexo.es)

Durante este mes vamos a continuar con la segunda parte del listado que dejamos pendiente. Correspondía al proyecto DictaPad, el cual soportaba la funcionalidad de efectuar dictado y leer las palabras reconocidas mediante el motor de texto hablado.

Seguimos el listado con el procedimiento Capitalizeword, el cual cambia de minúsculas a mayúsculas y viceversa la primera letra de una palabra seleccionada. Mediante el método FX lograremos capitalizar la primera letra. Para ello, es necesario establecer el valor 3 en su parámetro, tal y como se muestra en el listado:

```
Private Sub capitalizeword_Click()
 s = RichTextBox1.selstart
 e = RichTextBox1.SelLength
  Vdict1.Lock
  Vdict1.FX (3)
  Vdict1.Unlock
  RichTextBox1.selstart = s
  RichTextBox1.SelLength = e
End Sub
```

Los siguientes procedimientos corresponden al botón de comienzo de dictado de la barra de herramientas. Si se pulsa dicho botón, se establecerá o eliminará el modo escucha para el engine de dictado mediante la función Listen.

```
Private Sub listening_Click()
 listen (1)
End Sub
Private Sub notlistening_Click()
 listen (0)
End Sub
```

ReadDocument es la opción de menú que permite al ordenador leer el texto contenido en la caja Richtext. Esto es posible mediante la utilización del engine de texto hablado. Si no hay un engine de este tipo instalado en el ordenador, el procedimiento no tendrá efecto. El objeto que representa al engine de texto hablado es **DirectSS1**.

```
Private Sub ReadDocument_Click()
 Dim ReadText As String

 On Error Resume Next
```

Si al pulsar la opción de menú el engine está hablando, lo paramos mediante el procedimiento *Stopreading_Click*. Para comenzar a hablar, seleccionaremos el comienzo del texto y utilizaremos el método Speak del objeto **DirectSS1**.

```
If (DirectSS1.Speaking) Then
  Stopreading_Click
Else
  listen (0)
  Toolbar1.Buttons(15).Value = tbrPressed
  Toolbar1.Buttons(15).Image = 13
  gReadStart = RichTextBox1.selstart
  ReadText = Right$(RichTextBox1.Text,
      Len(RichTextBox1.Text) - RichText-
      Box1.selstart)
  DirectSS1.Speak ReadText
End If
End Sub

Private Function isTextBreak(c As String, bre-
      akchars As String)
For i = 1 To Len(breakchars)
  If c = Mid$(breakchars, i, 1) Then
    isTextBreak = True
    GoTo done
  End If
Next i
isTextBreak = False
done:
End Function
```

La función *Findtextbreak* se utiliza en conjunción con la función anterior Istextbreak para reducir el parpadeo *(flicker)* que se visualiza en la caja de

texto Richtext cuando se obtienen cadenas de caracteres. Para ello, es necesario buscar la longitud de cada palabra mediante funciones estándar de Visual Basic *Len* y *Mid$*. Estas funciones no son necesarias para el funcionamiento del programa, pero crean un mejor aspecto visual en la representación del texto obtenido.

```
Private Function FindTextBreak(direction As
        Integer, start As Integer, breakchars As
        String)
If (RichTextBox1.Text = "") Then
   FindTextBreak = 1
ElseIf (direction = 0) Then
   For i = start To 1 Step -1
      If (isTextBreak(Mid$(RichTextBox1.Text, i,
        1), breakchars)) Then
         FindTextBreak = i
         GoTo done
      End If

   Next i
   FindTextBreak = i
   GoTo done
Else
   For i = start To Len(RichTextBox1.Text)
      If (isTextBreak(Mid$(RichTextBox1.Text, i
        + 1, 1), breakchars)) Then
         FindTextBreak = i
         GoTo done
      End If
   Next i
   FindTextBreak = i
End If
done:
End Function
```

ReadDocument es la opción del menú que permite al ordenador leer el texto contenido en la caja Richtext

El siguiente procedimiento que vemos a continuación se ejecuta cuando se pulsa con el ratón sobre la caja de texto y normalmente se hace para seleccionar palabras. En la variable **breakstring** se definen una serie de caracteres que pueden delimitar la longitud de la palabra al ser encontrados:

```
Private Sub RichTextBox1_Click()
Dim breakstring As String

If (Alwaysselect.Checked) Then
   If RichTextBox1.selstart = Len(RichText-
        Box1.Text) Then GoTo setsel

   breakstring = " -_!@#$%^&*(),.[]+=`"""
        + vbNewLine + vbCrLf + vbCr + vbLf
   sStart = FindTextBreak(0,
        RichTextBox1.selstart, breakstring)
   If (sStart <> 0) Then
      If (sStart = RichTextBox1.selstart And
        (Not
        isTextBreak(Mid$(RichTextBox1.Text,
        (sStart), 1), breakstring))) Then
         sStart = sStart - 1
         sEnd = sStart + 1
      Else
         GoTo findit
      End If

   Else
findit:
      sEnd = FindTextBreak(1,
        RichTextBox1.selstart +
        RichTextBox1.SelLength, breakstring)
   End If
   RichTextBox1.selstart = sStart
   RichTextBox1.SelLength = sEnd - sStart
setsel:
   On Error GoTo done
```

Como se ha hecho anteriormente, para efectuar la selección de la palabra será necesario proteger y desproteger el objeto **Vdict1** mediante los métodos *Lock* y *Unlock*.

```
   Vdict1.Lock
   Vdict1.TextSelSet RichTextBox1.selstart,
        RichTextBox1.SelLength
   Vdict1.Unlock

done:
End If
End Sub
```

Cuando se selecciona texto es necesario definir la posición de la selección (método *TextSelSet*). *TextSet* cambia un texto seleccionado definido por el comienzo y el número de caracteres que tenga. El texto será reemplazado.

```
Private Sub SetText(newText As String, ui As
        Boolean)
 Vdict1.Lock
 Vdict1.TextSelSet RichTextBox1.selstart, 0
 Vdict1.TextSet newText, RichTextBox1.sels-
        tart, RichTextBox1.SelLength, 65536
 Vdict1.Unlock
If (ui) Then
 RichTextBox1.SelText = newText
End If
End Sub
```

La caja de texto que representará los caracteres obtenidos de las palabras del usuario debe incluir varios procedimientos que permitan seleccionar palabras mal reconocidas para poder editarlas posteriormente.

De este modo nosotros gestionaremos este tipo de eventos cuando se levante el dedo sobre el botón del ratón y cuando se presione alguna tecla *(MouseUp/Keypress)*.

```
Private Sub RichTextBox1_KeyPress(KeyAscii
        As Integer)
 Dim s As String
 s = Chr$(KeyAscii)
 SetText s, False
 ShowCorrectionWindow
        (showcorrection.Checked)
End Sub

Private Sub RichTextBox1_MouseUp(Button As
        Integer, Shift As Integer,
x As Single, y As Single)
 Vdict1.Lock
 Vdict1.TextSelSet RichTextBox1.selstart,
        RichTextBox1.SelLength
 Vdict1.Unlock
 ShowCorrectionWindow
        (showcorrection.Checked)
End Sub
```

Cuando se seleccione una palabra debe aparecer una ventana de corrección especializada. Para ello, visualizaremos este formulario mediante

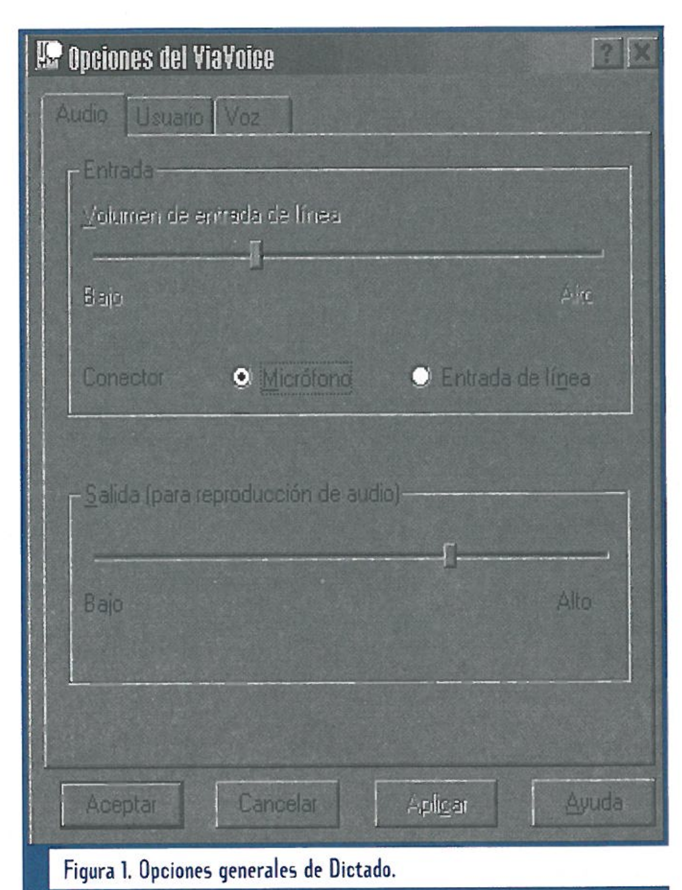


Figura 1. Opciones generales de Dictado.

ShowCorrectionWindow.

```
Private Sub showcorrection_Click()
If (showcorrection.Checked) Then
   ShowCorrectionWindow (0)
Else
   ShowCorrectionWindow (1)
End If
End Sub
```

Para parar la lectura del documento obtenido debemos aplicar al objeto **DirectSS1** el método *AudioReset*. Recordemos que **DirectSS1** es el objeto que representa al engine de texto hablado.

```
Private Sub Stopreading_Click()
 DirectSS1.AudioReset
End Sub
```

TextChanged es uno de los procedimientos más importantes del programa y se llama de forma automática cada vez que se cambia el texto por el objeto de dictado (no por el usuario). Si cuando obtiene texto obtiene palabras erróneas, intentará sustituirlas por otras que se asemejen bastante.

También será necesario proteger y desproteger el objeto **Vdict1** (métodos *Lock/ Un Lock*) para no incurrir en errores de protección y aplicar los cambios sin problemas.

```
Private Sub Vdict1_TextChanged(ByVal reason As Long)
Dim newStart As Long
Dim newend As Long
Dim oldStart As Long
Dim oldEnd As Long
Dim selstart As Long
Dim sellen As Long
Dim theText As String

Vdict1.Lock
On Error GoTo spuriouserror
```

Mediante el método *GetChanges* tomamos las diferencias y actualizamos la caja *Richtext* que visualiza las palabras:

```
Vdict1.GetChanges newStart, newend, oldStart,
      oldEnd
If (oldStart < oldEnd) Then
   RichTextBox1.selstart = oldStart
   RichTextBox1.SelLength = oldEnd - oldStart
   RichTextBox1.SelText = ""
End If

If (newend > newStart) Then
   RichTextBox1.selstart = newStart
   RichTextBox1.SelLength = 0
   Vdict1.TextGet newStart, newend - newStart,
      theText
   RichTextBox1.SelText = theText
End If
   ShowCorrectionWindow 2
   spuriouserror:
Vdict1.Unlock
End Sub
```

Open_Click es llamado cuando se carga un nuevo archivo de texto en la caja *RichText*. Podemos seleccionar tres tipos de archivos:

- TXT: archivo de texto
- MSD: archivo de Microsoft Dictation
- RTF: archivo de tipo Richtext

```
Private Sub open_Click()
 Dim versionstring As String
 Dim fn As String
 Dim fs As Long
 Dim fi As Long
 Dim fb As Long
 Dim fu As Long
 Dim fst As Long

 CommonDialog1.ShowOpen
 If (CommonDialog1.filename <> "") Then
  New_Click
  gThisFile = CommonDialog1.filename
```

El método *CreateStream* del objeto **Vdict1** se utiliza para crear un nuevo documento del tipo *MSD*. El segundo *flag* que se utiliza en sus parámetros debe ser **18**, ya que es el único valor aceptado por la *API* hasta el momento.

```
 If (Right(gThisFile, 3) = "msd") Then
   hand = Vdict1.CreateDocFile(gThisFile, 18)
   stream2 = Vdict1.CreateStream(hand,
    "Version", 18)
Vdict1.StreamRead stream2, versionstring, 36
```

Chequeamos si la versión del archivo se corresponde a un fichero *MSD*:

```
If versionstring <> "46FC730A-D849-11d0-
      AB8A-08002BE4E3B7" Then
dummy = MsgBox("Error. Formato inválido",
      vbOK, "Formato inválido")
```

En caso de no ser un archivo *MSD*, utilizaremos el método *CreateStream* para la carga del archivo. El primer parámetro dónde se almacenará el archivo (*handler*), el segundo representa el nombre de este *handler* para diferenciarlo. Por último asignaremos el valor **18** al último parámetro tal y como hemos hecho anteriormente.

```
Else
   stream = Vdict1.CreateStream(hand,
  "Header", 18)
   Vdict1.ReadStreamFont stream, fn, fs, fi,
  fb, fu, fst
   Vdict1.ReleaseStream stream
   Vdict1.SessionDeserialize hand
   RichTextBox1.Font.Name = fn
   If (fs < 0) Then
      fs = -fs
   End If
   RichTextBox1.Font.Size = fs
   RichTextBox1.Font.Italic = fi
   RichTextBox1.Font.Bold = fb
   RichTextBox1.Font.Underline = fu
  End If
```

Finalmente, vaciamos el *handler* utilizado y su objeto.

```
  Vdict1.ReleaseStream stream2
  Vdict1.ReleaseStore hand
 Else
  RichTextBox1.filename = CommonDia-
   log1.filename
  gThisFile = RichTextBox1.filename
  SetText RichTextBox1.Text, False
 End If
End If

End Sub
```

Para cambiar las opciones de dictado que permite el engine (cambio de usuario, sensibilidad de reconocimiento, etc.) utilizaremos el método *GeneralDlg*. En la siguiente imagen se puede apreciar el formulario que se visualizará al activar esta opción:

```
Private Sub Options_Click()
Vdict1.GeneralDlg Form5.hwnd, "Opciones de
      Dictado"
End Sub
```

Es necesario crear un procedimiento para guardar el archivo con el que estamos trabajando actualmente. Para ello, utilizaremos *DoSave*:

```
Private Sub DoSave()
Dim hand As Long
Dim fs As Long
Dim fi As Long
Dim fb As Long
Dim fu As Long
Dim fst As Long
```

Seguidamente crearemos el código para la grabación de los archivos *MSD*:

```
If (Right(gThisFile, 3) = "msd") Then
  fs = -RichTextBox1.Font.Size
  fi = RichTextBox1.Font.Italic
  fb = RichTextBox1.Font.Bold
  fu = RichTextBox1.Font.Underline
  fst = False

  hand = Vdict1.CreateDocFile(gThisFile, 18)
  stream = Vdict1.CreateStream(hand, "Hea-
     der", 18)
  Vdict1.SetSize stream, 0
  Vdict1.WriteStreamFont stream, RichText-
     Box1.Font.Name, fs, fi, fb, fu, fst
```

SessionSerialize guarda el estado actual del dictado al *handler* asignado (**hand**). La información incluirá todo el texto del buffer, el resultado de los objetos asociados con el texto y los marcadores. Una aplicación puede volver a cargar una sesión utilizando el método *SessionDeSerialize.*

```
  Vdict1.SessionSerialize hand
  stream2 = Vdict1.CreateStream(hand, "Ver-
     sion", 18)
  Vdict1.SetSize stream2, 0
  Vdict1.StreamWrite stream2, "46FC730A-
     D849-11d0-AB8A-08002BE4E3B7"
  Vdict1.ReleaseStream stream
  Vdict1.ReleaseStream stream2
  Vdict1.ReleaseStore hand
```

Creamos el código para la grabación de archivos *RTF*:

```
ElseIf (Right(gThisFile, 3) = "rtf") Then
  Open gThisFile For Output As 1
  Print #1, RichTextBox1.TextRTF
  Close 1
Else
```

Creamos el código para la grabación de archivos *TXT*:

```
  Open gThisFile For Output As 1
  Print #1, RichTextBox1.Text
  Close 1
End If
End Sub
```

El procedimiento *Listen* se utiliza para activar o desactivar la escucha del *engine* de dictado sobre el usuario. Mediante el método *Mode* se establece el estado de escucha utilizando alguno de estos *flags*:

- VSRMODE_DISABLED
  Dictado y command and control desactivados.
- VSRMODE_OFF
  Speech desactivado.
- VSRMODE_CMDPAUSED
  Speech en pausa (a la escucha para que un comando que lo despierte).
- VSRMODE_CMDONLY
  Command and control activado. Dictado desactivado.
- VSRMODE_DCTONLY
  Dictado activado. Command and control desactivado.
- VSRMODE_CMDANDDCT
  Los dos engines activados al mismo tiempo. Esto sólo es posible si el engine de reconocimiento lo permite.

```
Public Sub listen(op As Integer)

 If (op = 1) Then
       Stopreading_Click
       Vdict1.Mode = 32
 On Error GoTo NoActivate
       Vdict1.Activate
 NoActivate:
  Else
       Vdict1.Mode = 2
 On Error GoTo NoDeactivate
       Vdict1.Deactivate
```

```
NoDeactivate:

End If
ListenUI (op)
GoTo NoError
ErrorMessage:
 MsgBox "ERROR. Debe haber un engine ins-
     talado."
End
NoError:
End Sub
```

Finalizando el recorrido por los procedimientos más importantes del programa, llegamos al punto en el que nos encontramos con el procedimiento encargado de gestionar la barra de estado que contiene los botones de acceso rápido.

Podemos utilizar una propiedad denominada *Key de SelectCase* para poder así especificar la acción determinada que necesitamos realizar en cada momento.

```
Private Sub toolbar1_ButtonClick(ByVal Button
     As Button)
  Select Case Button.Key
  Case Is = "cut"
    Cut_Click
  Case Is = "cutall"
    SelectAll_Click
    Cut_Click
  Case Is = "copy"
    Copy_Click
  Case Is = "paste"
    Paste_Click
  Case Is = "new"
    New_Click
  Case Is = "save"
    save_Click
  Case Is = "open"
    open_Click
  Case Is = "listen"
    If (listening.Checked) Then
      listen (0)
    Else
      listen (1)
    End If
  Case Is = "showhide"
    showcorrection_Click
  Case Is = "capitalizeword"
    capitalizeword_Click
  Case Is = "addword"
    Addword_Click
  Case Is = "read"
    ReadDocument_Click
  End Select
End Sub
```

PRÓXIMAMENTE

En el siguiente número de esta serie de reconocimiento de voz veremos un programa sumamente útil de cara al usuario final.

Se trata de un programa que nos avisará, hablando, de los nuevos e-mails que hayamos recibido, diciéndonos el asunto del que trata el mensaje de viva voz.

HTML dinámico (II): El efecto scrolling

Adolfo Aladro (aaladro@arrakis.es)

En este segundo capítulo dentro de la serie sobre HTML dinámico vamos a profundizar en los contenidos que se presentaron a lo largo del artículo anterior. También se abordarán nuevas técnicas, entre las que destaca por ejemplo el efecto scrolling.

ÁREA DE CLIPPING (CLIPPING AREA)

Imaginemos por un momento que tenemos un cuadro y lo cubrimos con una tela oscura. Y ahora supongamos que esa tela oscura tiene un roto rectangular a través del cual podemos ver parte del lienzo. Pues bien, esa parte visible es lo que llamaríamos área de clipping.

De una manera más formal se consigue definir el área de clipping de una capa HTML como aquella zona que está visible, quedando oculto todo lo demás. El área de clipping se determina fijando cuatro atributos: top, bottom, left, right (ver Tabla 1).

Éstos pueden fijarse en el estilo de la capa o bien dinámicamente mediante JavaScript tal y como veremos más adelante. La figura 1 ilustra una capa y su área de clipping. Teniendo en cuenta la figura anterior es fácil deducir a simple vista que las distintas magnitudes involucradas se relacionan de la forma mostrada en la figura.

El método recortar es aquel que simplemente asigna los valores de los atributos del área de clipping diferenciando entre IE4 y NS4

Estas fórmulas debemos tenerlas presentes siempre ya que van a ser las que nos den las operaciones necesarias para llevar a cabo el efecto de scrolling.

EL EFECTO DE SCROLLING

¿Quién no ha visto alguna vez los créditos finales de una película que salen por la parte inferior de la pantalla y van desplazándose hacia arriba? Pues eso es ni más ni menos un scrolling. Este efecto es el que vamos a tratar de lograr sacando provecho del área de clipping de las capas. El truco consiste en mantener el área de clipping fija en la misma posición mientras que la capa va desplazándose. Así se crea la ilusión de que el área de clipping es una ventana por la que va pasando la capa (figura 2).

La posición del área de clipping viene dada por el punto (xc,yc), y este punto depende a su vez de los valores que tomen los atributos top, bottom,

TABLA 1. Atributos que definen el área de *clipping* de una capa.

top	Distancia que hay entre el extremo superir de la capa y el extremo superior del área de clipping.
bottom	Distancia que hay entre el extremo superior de la capa y el extremo inferior del área de clipping
left	Distancia que hay entre el extremo lateral izquierdo de la capa y el extremo lateral izquierdo del área de clipping
right	Distancia que hay entre el extremo lateral izquierdo de la capa y el extremo lateral derecho del área de clipping

right y left. Si queremos mantener el punto (xc,yc) fijo mientras la capa se desplaza, habrá que ir calculando los valores de top, bottom, left y right en cada momento en función de la posición de la capa (x,y). En concreto sólo serán los valores de top y bottom ya que los atributos relativos al ancho se mantendrán constantes a efectos del scrolling que queremos realizar.

(x0,y0) (x1,y1) ··· (xi,yi) ··· (xn,yn)

Antes de que comience el scrolling tendremos que:

yc = y0

Teniendo en cuenta que xc, yc y heightc son constantes, esto implica que los valores iniciales que han de tomar los atributos relativos al área de clipping son los siguientes:

1) $y_c = y_i + top_i$ $\quad y_c = y_0 + top_0$
   $top_0 = y_c - y_c$ $\quad \mathbf{top_0 = 0}$

2) $height_c = bottom_c - top_i$
   $height_c = bottom_0 - 0$
   $\mathbf{bottom_0 = height_c}$

Por cada movimiento de la capa los valores de top y bottom serán:

1) $y_c = y_i + top_i$ $\quad \mathbf{top_i = y_c - y_i}$

2) $height_c = bottom_i - top_i$
   $\mathbf{bottom_i = height_c + top_i}$

La capa habrá hecho todo el scrolling cuando ésta se sitúe totalmente fuera del área del clipping (figura 3). Esto se producirá cuando:

topi >= height

Ahora hay dos opciones: detener el scrolling o volver a empezar. El scrolling continuo se produce cuando la capa vuelve a aparecer por la zona inferior del área visible inmediatamente después de desaparecer por la parte superior. Una vez producida la condición de terminación descrita hay que mover la capa de forma que al continuar con el scrolling ésta aparezca por el extremo inferior de la zona visible. Calculamos esta posición teniendo en cuenta que el comienzo del scrolling por la zona inferior del área visible viene dado por la expresión:

bottomi = height

Lo que implica:

1) $height_c = bottom_i - top_i$
   $\mathbf{top_i = height + height_c}$

2) $y_c = y_i + top_i$
   $\mathbf{y_i = y_c + top_i = y_c + height + heigh_c}$

Una vez que hemos se ha descrito la teoría, convertirla a JavaScript es fácil partiendo del script que realizamos en el artículo anterior. El Listado 1 muestra nuestro nuevo método objeto-Capa donde resaltamos los métodos y las propiedades que hemos añadido.

DIFERENCIAS ENTRE IE4 Y NS4 AL DEFINIR EL ÁREA DE CLIPPING

En primer lugar debemos observar que hemos añadido una nueva propiedad

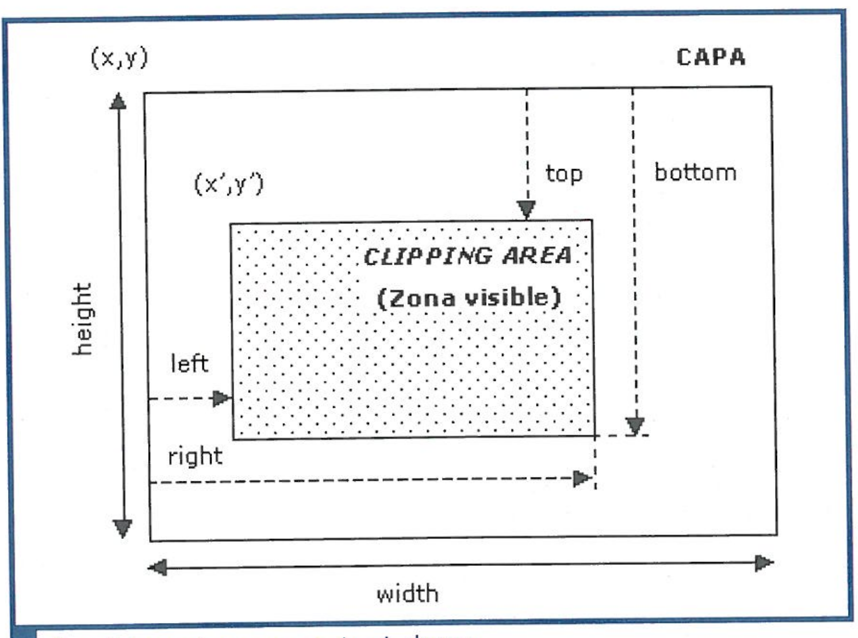


Figura 1. Imagen de una capa y su área de clipping.

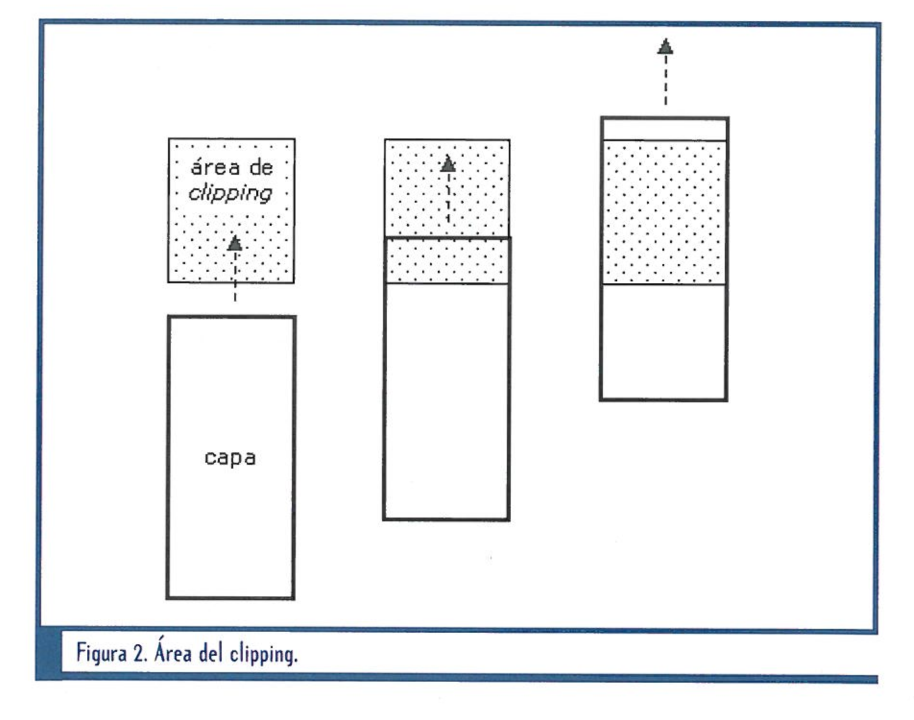


Figura 2. Área del clipping.

llamada recorte que no es más que un array de cuatro elementos en el que se guardarán los valores de top, bottom, left y right. IE4 y NS4 tienen formas completamente distintas de acceder a los atributos relativos al clipping. NS4 puede acceder individualmente a cada uno de estos valores haciendo uso de las propiedades clip.top, clip.bottom, clip.left y clip.right de la capa.

IE4, por el contrario, accede mediante una única propiedad llamada clip que ha de contener una cadena del tipo rect(valor_top valor_right valor_bottom valor_left). Esto hace ciertamente engorrosa la tarea de acceder individualmente a una sola de las propiedades ya que tenemos que buscar en la cadena el valor de propiedad deseado. Por ello hemos decidido crear y mantener el array recorte de tal forma en que cada posición del mismo estarán en todo momento los valores de los atributos del área de clipping tanto para NS4 como para IE4.

El método recortar es aquel que simplemente asigna los valores de los atributos del área de clipping diferenciando entre IE4 y NS4 tal y como hemos explicado. Al mismo tiempo se actualizan los elementos del array recorte.

El método scrolling sirve para asignar al objeto capa propiedades que posteriormente utilizaremos cuando efectivamente hagamos el scrolling. Se puede ver a simple vista que todas ellas se correponden con valores de los que hemos hablado al explicar los fundamentos teóricos del efecto de scrolling.

Finalmente el método scrollingUp es el encargado de llevar a cabo el efecto. Simplemente se trata de aplicar las fórmulas descritas anteriormente. Al igual que ocurría con las animaciones, utilizamos la función setTimeout la cual es llamada cada cierto tiempo para actualizar todos los valores del scrolling.

CREACIÓN DINÁMICA DE ESTILOS

Hasta ahora hemos creado las capas utilizando estilos que definíamos mediante la etiqueta STYLE al principio del documento HTML. Pero, ¿qué ocurre si por ejemplo el ancho de la capa que queremos crear es una función del ancho de la ventana del navegador?

Quizás lo primero que se nos ocurre es que podríamos crear la capa con un ancho determinado y cambiarlo con JavaScript más tarde. Bueno, esto es cierto para muchos atributos pero desgraciadamente no lo es para todos, y en general depende del navegador.

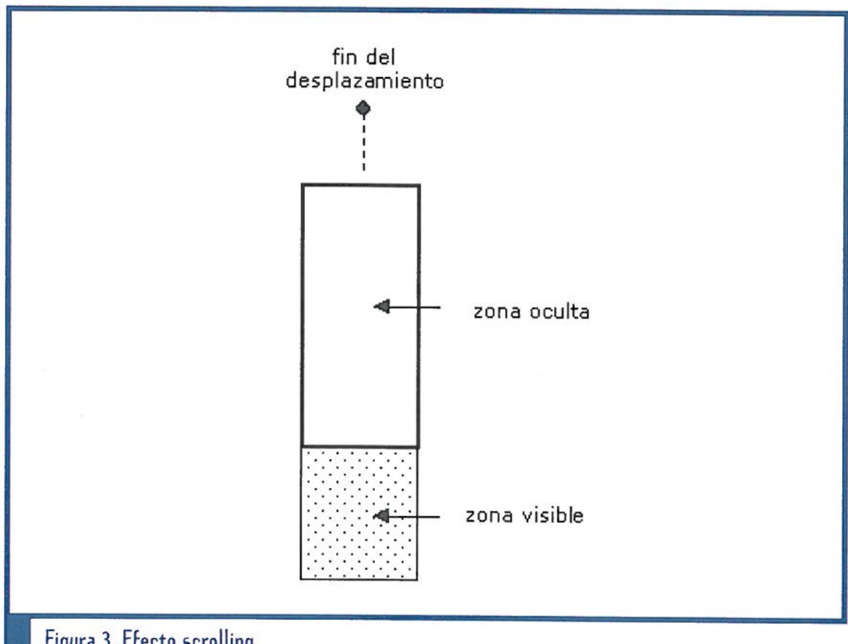


Figura 3. Efecto scrolling.

WWW

En otras palabras, hay casos en los que es preciso que haya una serie de atributos en la propia definición del estilo, ya que luego no los podremos modificar. Además aunque se trate de un atributo que se pueda modificar, puede ser necesario que la capa se inicialice de una forma determinada para que no se produzcan efectos extraños de acople entre que la capa se visualiza y se ajustan sus atributos.

La solución a todos estos problemas es la creación dinámica de estilos, que se trata sencillamente de generar la etiqueta STYLE, con todas las definiciones de estilos, utilizando JavaScript.

Supongamos que tenemos el siguiente estilo definido al principio de un documento HTML:

```
<HTML>
<HEAD>
<STYLE TYPE="text/css">
.miestilo {
  position: absolute;
  top: 0px;
  left: 0px;
}
</STYLE>
</HEAD>
<BODY>
...
</BODY>
</HTML>
```

Pues ese mismo estilo podríamos generarlo de forma dinámica haciendo:

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!—
function generarEstilo() {
  codHTML =  '<STYLE TYPE="text/css">';
  codHTML += '.miestilo {';
  codHTML += 'position: absolute;';
  codHTML += 'top: ' + pos_top_inicial + 'px;';
  codHTML += 'left: ' + pos_left_inicial + 'px;';
  codHTML += '}';
  codHTML += '</STYLE>';
  document.write(codHTML);
}
```

Listado 1. Constructor del objeto capa.

```
function objetoCapa(nombreCapa) {

  objeto = eval(layerObj + '["' + nombreCapa + '"]' + styleObj);

  objeto.nombre = nombreCapa;
  objeto.mover = mover;
  objeto.mostrar = mostrar;
  objeto.ocultar = ocultar;
  objeto.animacion = animacion;

  objeto.recortar = recortar;
  objeto.recorte = new Array(4);
  objeto.recorte["top"] = -1;
  objeto.recorte["right"] = -1;
  objeto.recorte["bottom"] = -1;
  objeto.recorte["left"] = -1;

  objeto.scrolling = scrolling;
  objeto.scrollingUp = scrollingUp;

  objeto.getWidth = getWidth;
  objeto.getHeight = getHeight;

  objeto.actualizarContenido = actualizarContenido;

  coleccionObjetosAnimados[nombreCapa] = objeto;
  return objeto;
}
```

Listado 2. Método recortar del objeto capa.

```
function recortar(ntop, nright, nbottom, nleft) {
var cadena='';
  if (NS4) {
    eval(layerObj + '["' + this.nombre + '"]' + styleObj + '.clip.top = ntop');
    eval(layerObj + '["' + this.nombre + '"]' + styleObj + '.clip.right = nright');
    eval(layerObj + '["' + this.nombre + '"]' + styleObj + '.clip.bottom = nbottom');
    eval(layerObj + '["' + this.nombre + '"]' + styleObj + '.clip.left = nleft');
  } else {
    cadena += 'rect(' + ntop + ' ' + nright + ' ' + nbottom + ' ' + nleft +')';
    eval(layerObj + '["' + this.nombre + '"]' + styleObj + '.clip = cadena');
  }
  this.recorte["top"] = ntop;
  this.recorte["right"] = nright;
  this.recorte["bottom"] = nbottom;
  this.recorte["left"] = nleft;
}
```

```
pos_top_inicial = 100;
pos_left_inicial = 100;
generarEstilo();
//-->
</SCRIPT>
</HEAD>
<BODY>
...
</BODY>
</HTML>
```

Como vemos la función generar Estilo simplemente construye una cadena con el código correspondiente al estilo de la capa, lo que nos permite introducir variables que fijen dinámicamente los atributos. Una vez que la cadena está construida se escribe en el documento utilizando el método write del objeto document.

Nótese que es imprescindible que la llamada a generarEstilo se haga antes de que el navegador haya empezado a "parsear" la parte correspondiente a la etiqueta BODY.

Listado 3. Métodos relacionados con el scrolling de la capa.

```
function scrolling(scrollLoop, scrollSpeed, scrollPixels, scrollAreaWidth, scrollArea-
Height) {
  this.scrollLoop = scrollLoop;
  this.scrollSpeed = scrollSpeed;
  this.scrollPixels = scrollPixels;
  this.scrollAreaWidth = scrollAreaWidth;
  this.scrollAreaHeight = scrollAreaHeight;
  this.scrollTop = parseInt(this.top,10);
  this.recortar(this.scrollTop, scrollAreaWidth, scrollAreaHeight, 0);
  this.finScroll = finScroll;
  this.scrollPause = false;
}

function scrollingUp() {
  if (!this.scrollPause) {
    if (this.recorte["top"] < this.getHeight()) {
      this.top = parseInt(this.top,10) - this.scrollPixels;
      recorteTop = this.scrollTop - parseInt(this.top,10);
      recorteBottom = this.scrollAreaHeight + recorteTop;
      this.recortar(recorteTop, this.recorte["right"], recorteBottom, this.recorte["left"]);
      setTimeout('coleccionObjetosAnimados["'+ this.nombre + '"].scrollingUp()',
this.scrollSpeed);
    } else {
      if (this.scrollLoop > 0) {
        this.top = this.scrollAreaHeight;
        recorteTop = - this.scrollAreaHeight;
        recorteBottom = 0;
        this.recortar(recorteTop, this.recorte["right"], recorteBottom, this.recorte["left"]);
        this.scrollLoop--;
        setTimeout('coleccionObjetosAnimados["'+ this.nombre + '"].scrollingUp()',
this.scrollSpeed);
      } else {
        this.finScroll();
      }
    }
  }
}
```

PROPIEDAD WIDTH: ANCHO Y ALTO DE LA VENTANA DEL NAVEGADOR

IE4 y NS4 tienen formas distintas de tratar con la propiedad width del estilo una capa. Para empezar, si no especificamos nada, NS4 asume por defecto que width toma el ancho que realmente ocupa la capa, mientras que IE4 hace que la capa tome todo el ancho disponible en la ventana del navegador.

Por otro lado, NS4 necesita saber el valor del atributo width de un elemento antes de que éste sea creado, es decir, debe aparecer en la definición del estilo. Con IE4 podemos asignar un valor a la propiedad width en cualquier momento.

Puede ser frecuente que queramos asignar un valor a la propiedad width dependiendo del ancho que tengamos disponible de la ventana del navegador, el cual podemos obtener fácilmente:

IE4	document.body.clientWidth
NS4	window.innerWidth

Aunque en principio *document.body.clientWidth* y *window.innerWidth* son equivalentes, existe una peculiaridad relevante que los va a diferenciar. Estas dos propiedades son iguales únicamente en el caso de que el cuerpo de la página haya empezado a cargarse. NS4 basa su propiedad en la ventana (window), la cual existe antes de que la página se cargue. Por el contrario, IE4 basa su propiedad en el cuerpo de la página (document.body), y este valor no lo podemos conocer hasta que la página ha sido cargada correctamente.

Si le gustaron otras bases de datos, 4th Dimension v6 le sorprenderá

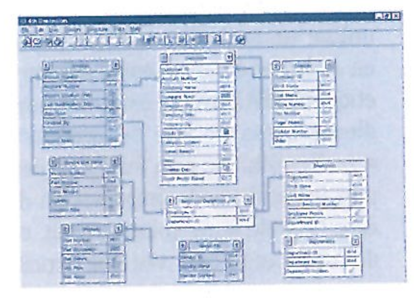


4th Dimension es el entorno de bases de datos integrado más completo del mercado. Años de experiencia en el diseño de bases de datos han permitido satisfacer las siguientes necesidades:

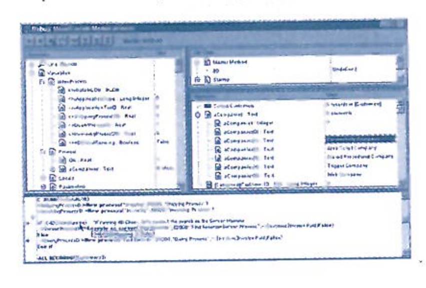
- Automatizar cualquier proceso.
- Diseñar modelos gráficos eficientes y orientados a objeto.
- Suministrar la más amplia gama de herramientas para facilitar el diseño y mantenimiento de bases de datos.
- Permitir una fácil reutilización de código.
- Permitir escalar desarrollos Cliente/ Servidor, web y multiplataforma de forma transparente.

Estas son algunas de las **características** de 4th Dimension:

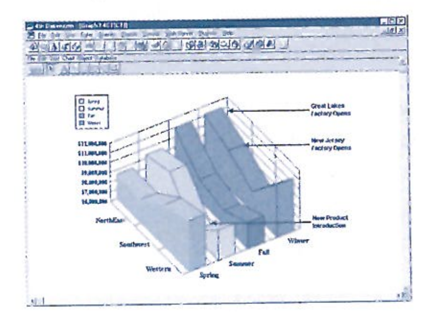
- **Motor de base de datos multiproceso, triggers, procedimientos almacenados**, presentación visual de tablas mediante diagrama de entidad /relación (ERD)



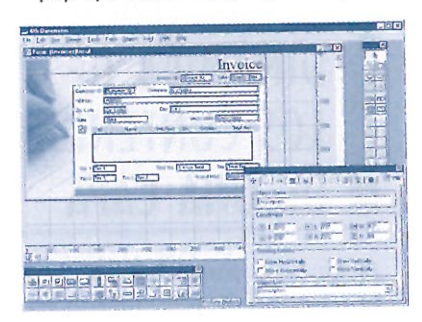
- **Debugger visual** multiproceso de última generación. **Intérprete** incluido para testeo y debugging.



- **Wizards avanzados** para diseño de formularios, queries, informes, importación, exportación, etiquetas, etc. Editor de gráficas programable 2D y 3D, soporta gráficas de barras, tartas, áreas, líneas, etc.



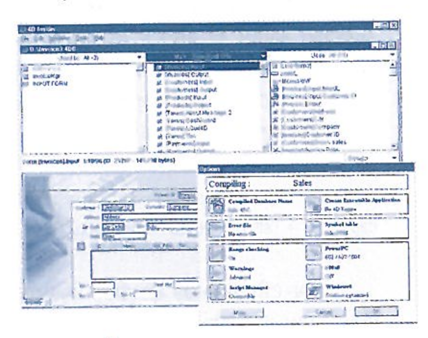
- **Editor de formularios basado en objetos.** Gestión de hojas de estilo. Redimensionamiento automático. **Docenas de objetos de interface** incluyendo check boxes, radio buttons, push buttons, áreas de scroll, picture button, botones invisibles, tab control, listas jerárquicas, termóme-tros, reglas, combo box, listas menú desplegables, menús pop-up jerárquicos, menús picture, radio pictures, listas desplegables, menús pop-up, subforms, group boxes, etc.



- Arquitectura de distribución de aplicaciones integrada y **soporte para proyectos con múltiples desarrolladores** con check-in/check-out automatizado.
- Arquitectura extensible mediante **plug-ins** y DLLs.
- **Un sólo código fuente** desplegable sin modificaciones de monousuario a 4D Server, la versión Cliente/Servidor de 4th Dimension.
- Soporte de **ODBC** (cliente y servidor) y conexiones nativas **SQL** a ORACLE, Sybase y MS SQL Server.
- **Servidor Web integrado** para acceso automático a formularios y gestión Cliente/Servidor mediante navegadores web.



- **Compilador a código máquina real** optimizado para diferentes procesadores. **Gestor gráfico de referencias cruzadas** y librerías de objetos.



VERSIÓN DEMO GRATUITA

4th Dimension es el núcleo de la gama de productos 4D. Visite http://www.ambitconsulting.com o llame al 93-209 41 11 para más información o para recibir una versión demo gratuita. Mencione este anuncio y reciba 4th Dimension v6 por el precio promocional de 34.900 Ptas*. (precio habitual: 79.000 Ptas.).



AMBIT Consulting

Balmes, 297, 4º, 2ºb. 08006 Barcelona
t. | (34) 93 209 41 11 f. | (34) 93 240 18 88
e-mail: ambit@ambitconsulting.com
http://www.ambitconsulting.com



Bellsoft, S.L. C/Salva, 4-2º-2ª 46002 - Valencia
t. | (34) 96 352 71 30 f. | (34) 96 352 71 30
AppleLink SPA0351 e-mail: bellsoft@combios.es
http://www.bellsoft.es

(* Actualización competitiva. Oferta válida hasta el final de existencias)

Listado 4. Métodos para obtener el ancho (getWidth) y el alto (getHeight) de una capa.

```
Function getWidth() {
  widthPropertie = (NS4) ? '.document.width' : '.offsetWidth';
  currentWidth = eval('parseInt(' + layerObj + '["'+ this.nombre + '"]' + widthPropertie +
', 10)');
  return currentWidth;

}

function getHeight() {
  heightPropertie = (NS4) ? '.document.height' : '.offsetHeight';
  currentHeight = eval('parseInt(' + layerObj + '["'+ this.nombre + '"]' + heightPropertie
+ ', 10)');
  return currentHeight;
}
```

EL ANCHO Y EL ALTO DE UNA CAPA

Cuando vamos a posicionar (posicionamiento absoluto) una capa en el documento HTML muchas veces es preciso saber el alto y el ancho de la misma a fin de colocarla correctamente. Por ejemplo, para centrar horizontal y verticalmente haremos:

```
capa.left = (ancho_ventana – ancho_capa) / 2;
capa.top  = (alto_ventana – alto_capa) / 2;
```

Por ello hemos añadido dos métodos a nuestro objeto capa que nos proporcionan el alto y el ancho de capa. IE4 y NS4 tienen formas distintas de proporcionarnos estos valores y como siempre los métodos sirven para ocultar estas diferencias (ver Listado 4). Sea cual sea el navegador la llamada a los métodos getWidth o getHeight nos devolverá el ancho y alto respectivamente de la capa.

ACTUALIZACIÓN DINÁMICA DE LOS CONTENIDOS DE UNA CAPA

Otra de las posibilidades de las capas es el hecho de poder cambiar dinámicamente su contenido. IE4 y NS4 tienen formas muy distintas de hacerlo. Como siempre, hemos definido un nuevo método que se encarga de la actualización de los contenidos de una capa (Listado 5).

NS4 asocia a cada capa un objeto document el cual es accesible mediante la propiedad document de la capa. Este objeto es similar al objeto window. document que utilizamos con frecuencia. Cuenta entre otros con tres métodos que nos permiten cambiar su contenido: open abre el documento (La sola llamada a este método hace que automáticamente se pierda el contenido anterior); write("…") nos sirve para escribir los nuevos contenidos; y close cierra el documento.

Teniendo en cuenta esto podría parecer a primera vista que todo muy sencillo pero la cosa se complica cuando trabajamos con estilos. Por alguna razón NS4 pierde el estilo cuando actualizamos el documento asociado a una capa.

Para evitar que esto ocurra podemos utilizar un pequeño truco que consiste en embeber el nuevo contenido en una etiqueta SPAN con el estilo correspondiente antes de que se produzca la actualización. De esta forma no perdemos los atribtos de presentación de la capa (tipo de letra, color, tamaño, color de fondo, etc.) cada vez que actualizamos su contenido.

Con IE4 la actualización resulta mucho más sencilla. En primer lugar IE4 no pierde el estilo con las actualizaciones por lo que no es necesaria la utilización de la etiqueta SPAN.

Por otro lado, existe una propiedad llamada innerHTML que que contiene el código HTML asociado a la capa. Esta propiedad es tanto de lectura como de escritura.

Por lo tanto, nos basta con actualizar la propiedad innerHTML de la capa para actualizar el contenido de la misma.

Listado 5. Métodos para actualizar el contenido de una capa.

```
Function actualizarContenido(contenido) {
  if (NS4) {
    eval(layerObj + '["' + this.nombre + '"]' + styleObj + '.document.open()');
    eval(layerObj + '["' + this.nombre + '"]' + styleObj + '.document.write(contenido)');
    eval(layerObj + '["' + this.nombre + '"]' + styleObj + '.document.close()');
  } else {
    eval(layerObj + '["' + this.nombre + '"].innerHTML = contenido');
  }
}
```



Algunos Ratones son muy curiosos...

INFORMACIÓN

¡Proteja la información de sus aplicaciones!

DadvinaFence
Seguridad en sus Aplicaciones.

Uno de los riesgos que sufren las aplicaciones informáticas es la posibilidad de su uso por personas no autorizadas.
DadvinaFence incorpora tres componentes OCX o VCL:**ApplicationLock**, **FormLock** y **NetMessenger**, proporcionando al programador un sistema de seguridad activo fácil de integrar y totalmente configurable.

- Versiones VCL para Delphi 3.0/2.0, C++ Builder 3 y OCX para Visual Basic 5.0.
- Asignación de permisos y privilegios sobre los objetos existentes en las pantallas de las aplicaciones.
- Autentificación de usuarios mediante el uso de identificación y contraseña.
- Detección de intentos fallidos continuados de entrada al sistema,avisando al instante a los administradores de seguridad de las aplicaciones.
- Centralización de los accesos (login) a las bases de datos de la aplicación.
- Registro de entradas y salidas.
- Sistema modular fácilmente actualizable en tiempo de ejecución y de diseño.

etc...



Sus clientes se lo Agradecerán

Sólo por 31.500* ptas.



San Toribio 6.
28031 Madrid.
Email: com@dadvina.com

Para más información o realizar pedidos contacte con el telefono 91 380 32 46 o en nuestra Web: www.dadvina.com.

* IVA no incluido.

Herramientas para construir un cortafuegos (y III)

Juan José Taboada León (taboada@uhu.es) y José Juan Mora Pérez (albeniz@uhu.es)

Tal como quedó pendiente en el artículo anterior, en estas páginas trataremos una solución software más completa orientada a la construcción de un cortafuegos. En este caso y para poderlo llevar a cabo se utilizará una herramienta de gran ayuda: *FireWall Toolkit.*

Hemos elegido *Fwtk* (**TIS FireWall Toolkit**) de la empresa *Trusted Information Systems*, que se dedica al desarrollo de herramientas de seguridad debido a dos causas. La primera se debe a que esta herramienta es de carácter gratuito, lo que permitirá evaluarla sin ningún tipo de restricción.

La espina dorsal de Fwtk es el fichero netperm-table

Y la segunda razón, si cabe aun más importante, es que se distribuye con el código fuente, con lo cual tendremos una gran oportunidad para examinar cómo trabaja este tipo de software. Debido a su relativa popularidad la podemos encontrar en muchos servidores de *FTP*, y últimamente suele estar incluida en algunas distribuciones de Linux. Herramienta incluida en el CD-ROM.

FIREWALL TOOLKIT

Esta herramienta ha sido diseñada para configurar un cortafuegos como máquina Bastión, ya que el filtrado de paquetes, depende de que nuestro sistema se encuentre o nó configurado para "rutar" tráfico.

El paquete *FireWall Toolkit* consta de varios elementos, estos son los siguientes:

- Servidores *Proxies* para *TELNET*, RLOGIN, *FTP*, *HTTP*, *GOPHER*, *X*, *SMTP* y *NNTP*.
- *Netacl*, es un programa de control de acceso, parecido a *TCP Wrapper*.
- Herramientas de administración y auditoría tales como:

  - *Portscan.* Permite realizar un análisis sobre los puertos abiertos en una máquina.
  - *Netscan.* Chequea una red para ver que máquinas se encuentran en funcionamiento.
  - *Fwtk* ofrece una serie de *scripts* que generan informes sobre el uso de algún servicio determinado.
- Un servidor de autentificación.
- Clientes y servidores de algunos servicios. Dentro de los últimos podemos encontrar un servidor

Tabla 1. Reglas de TN-GW.	
Authserver hostname puerto	Nombre o dirección de un servidor de autentificación y el puerto.
denial-msg nombre-fichero	Nombre del fichero que se visualizará en caso de que no se acepte la conexión
Directory nombre-directorio	Nombre del directorio que tn-gw utilizará como raíz, mediante chroot.
deny-hosts	Especifica los hosts que tienen denegado el acceso.
help-msg	Nombre del fichero que se visualizará en caso de que un usuario pida ayuda.
permit-host	Especifica los hosts que tienen permiso de acceso
Timeout segundos	Especifica el número de segundos que debe esperar tn-gw antes de cerrar la conexión

de *FTP* y un sustituto para el programa *login* de nuestro sistema. Ambos hacen uso del servidor de autentificación proporcionado por *Fwtk*.

La espina dorsal de *Fwtk* es el fichero **netperm-table**, cuya localización depende de cómo lo hayamos configurado a la hora de compilar el código fuente de la herramienta (desde ahora vamos a suponer que se encuentra en /etc). En este fichero definiremos la política de seguridad que va a seguir nuestro cortafuegos.

Este fichero se encuentra dividido en varias secciones, cada una de las cuales esta constituida por una serie de reglas, cuya misión consiste en definir el comportamiento de los distintos elementos de *Fwtk*, ya sea el *Proxy TELNET*, *FTP*, el servidor de autentificación, etc.

NETACL

Se trata del software encargado de realizar una filtración, exactamente igual a como la hace *TCP Wrapper*. Para comprender la forma en que trabaja *netacl* estudiaremos un ejemplo práctico.

Queremos que únicamente las máquinas cuyas IP pertenezcan al dominio **111.112.113.0** puedan acceder al servicio *TELNET* de nuestro sistema. En primer lugar tendremos que modificar en el fichero **/etc/inetd.conf** la línea dedicada a la gestión de las conexiones *TELNET*.

```
telnet stream tcp nowait root /usr/sbin/telnet
```

Y la sustituiremos por la línea siguiente:

```
telnet stream tcp nowait root /usr/sbin/netacl
        telnet
```

Esta herramienta ha sido diseñada para configurar un cortafuegos como máquina Bastión

Netacl sólo acepta un parámetro, con el cual construye una cadena con el formato **netacl-parametro**, (en nuestro ejemplo la cadena consistiría en **netacl-telnet**). Esta cadena es utilizada por *netacl* para encontrar las reglas en el fichero **/etc/netperm-table**, que como ya hemos mencionado antes contiene las reglas que se aplicarán cada vez que se realice una petición de conexión *TELNET*, hacia o desde el cortafuegos, utilizando el puerto **23**, que es el standard para este servicio.

Las reglas de **/etc/netperm-table** para *netacl* son del tipo:

```
netacl-telnet: permit-hosts 111.112.* -exec
        /usr/etc/in.telnetd
```

El primer campo es el nombre de la regla, el siguiente la regla que se aplica a la conexión y por último, la acción que se realizará en caso de que la regla sea válida.

La regla anterior, permite todas aquellas conexiones *TELNET* que se realicen desde una máquina cuya IP pertenezca a la familia **111.112.*.***. Si la regla es válida se ejecuta **in.telnetd** que se encargará de tratar la conexión.

La espina dorsal de Fwtk es el fichero netperm-table

Podríamos crear una regla como la siguiente, la cual prohibe el acceso a nuestra máquina a través de *TELNET* y en caso de que se produzca un intento de conexión, se visualiza el contenido del fichero **aviso.txt**:

```
netacl-telnet: deny-hosts * -exec /usr/bin/cat
        aviso.txt
```

Básicamente ya hemos visto la forma en que trabaja **netacl**, destacando por la creación de las reglas apropiadas para realizar una filtración lo más fina posible de las conexiones que se realicen con nuestro sistema.

Seguidamente vamos a efectuar un breve recorrido por los distintos elementos de los que dispone *Fwtk*.

TN-GW

Este es el servidor *Proxy* para *TELNET* que ofrece *TIS* con *Fwtk*. Ya conocemos la forma en la que nuestro sistema UNIX opera cada vez que recibe una petición de conexión, por lo tan-

Listado 1. Reglas para tn-gw y ftp-gw.

```
#
# Reglas para el servidor Proxy TELNET
#
tn-gw: authserver localhost 2048
tn-gw: welcome-msg /usr/local/fwtk/tn-wel.txt
tn-gw: denial-msg /usr/local/fwtk/tn-den.txt
tn-gw: help-msg /usr/local/fwtk/tn-hel.txt
tn-gw: timeout 1000
tn-gw: permit-hosts 111.112.* -dest 123.123.*
tn-gw: permit-hosts 111.113.* -auth
tn-gw: deny-hosts *

#
# Reglas para el servidor Proxy FTP
#
ftp-gw: welcome-msg /usr/local/fwtk/ftp-wel.txt
ftp-gw: denial-msg /usr/local/fwtk/ftp-den.txt
ftp-gw: help-msg /usr/local/fwtk/ftp-help.txt
ftp-gw: timeout 1000
ftp-gw: permit-hosts 111.112.* -dest 190.1.1.* -deny stor
ftp-gw: permit-hosts 111.112.* -dest 123.123.*
ftp-gw: deny-hosts *
```

to a la hora de implantar un servidor *Proxy* para el servicio de *TELNET*, lo primero que debemos hacer es modificar el fichero **/etc/inetd.conf**. Así, la línea encargada de tratar las conexiones *TELNET* la modificaremos para que el elemento encargado de tratar la conexión no sea el sistema UNIX, sino el software ofrecido por *Fwtk*.

```
telnet stream tcp nowait root /usr/sbin/
    tn-gw tn-gw
```

De esta forma cuando **inetd** reciba una petición de conexión por el puerto **23** ejecutará el programa **/usr/sbin/tn-gw**, el cual, cargará la sección correspondiente al parámetro **tn-gw** del fichero **/etc/netperm-table**.

Está sección está constituida por reglas, las cuales utilizan una serie de cláusulas que se muestran en la tabla 1.

En el listado 1 se puede apreciar un ejemplo de esta sección, con las reglas para *tn-gw* y *ftp-gw*.

El orden en el que se especifican las reglas es el orden en el que se aplican

Las tres primeras reglas del listado 1 (vamos a ignorar la primera línea, la cual está relacionada con el servidor de autentificación) especifican los ficheros que se visualizarán en caso de que se realice una conexión con éxito, denegada o se solicite ayuda respectivamente.

Las dos últimas son las reglas de acceso, que cuentan con varias opciones:

- **dest** *IP*, podemos condicionar una regla según su destino, lo que permitirá controlar hacia dónde se realizan las peticiones de conexión desde nuestro sistema (esto puede ser bastante útil a la hora de controlar el acceso a ciertos servidores).
- **auth**, el *Proxy* solicita autentificación al usuario que lo usa.
- **passok**, el *Proxy* debe permitir que los usuarios cambien sus contraseñas.
- **permit** {*operación1 operación2* ...}, el *Proxy* sólo permite que se realicen la lista de operaciones, la cual cambia, según el servicio sobre el que trabaja el *Proxy*. Por ejemplo para un servidor *Proxy FTP*, las operaciones válidas serían las permitidas por el demonio **ftpd**, tales como:
  - **stor**, almacenar un fichero.
  - **retr**, recuperar un fichero.
  - **rmd**, eliminar un directorio.
  - **pwd**, ver cuál es el directorio actual.
  - **list**, listado de los ficheros del directorio actual.
  - **deny** {*operación1 operación2* ...}, es exactamente igual que la anterior con la diferencia de que ésta deniega las operaciones listadas.
  - **log** {operación1 operación2 ...}, registra todas las operaciones listadas.

Estos son algunos ejemplos de los parámetros que puede incluir una regla. Cada servicio, ya sea *TELNET*, *FTP*, etc., tiene su conjunto de operaciones, las cuales podremos utilizar con estos parámetros.

Las reglas del listado 1 expresan la siguiente política:

- Sólo se permite el acceso al *Proxy* a aquellas máquinas cuya IP pertenezca a la familia **111.112.*** y quieran acceder a máquinas dentro del dominio **123.123.***.

- Se deniega el acceso a todas la demás.

El orden en el que se especifican las reglas es el orden en el que se aplican, lo cual es bastante importante, ya que si hubiéramos intercambiado las posiciones de las dos ultimas reglas, el resultado hubiera sido, que no se permitiría el acceso a nadie. Por lo tanto tenemos que tener extremo cuidado a la hora de crear un conjunto de reglas amplio.

Lo más importante de netacl es la creación de las reglas apropiadas para realizar una filtración

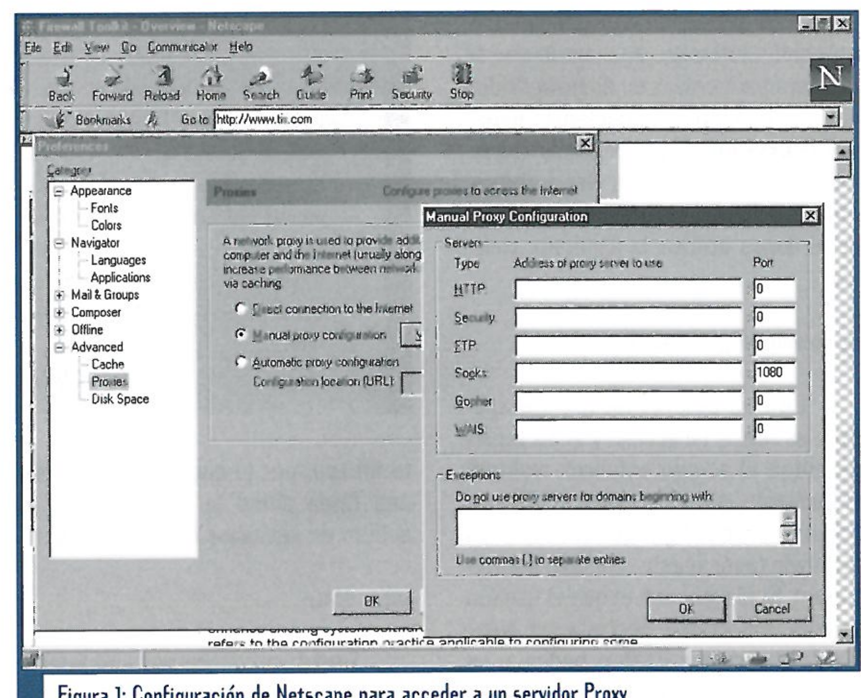


Figura 1: Configuración de Netscape para acceder a un servidor Proxy.

Una vez que hayamos configurado el servidor *Proxy* para atender el servicio de *TELNET*, cualquier máquina situada en nuestra red, para atravesar el cortafuegos, debe realizar primero una conexión *TELNET* con nuestro sistema UNIX que será tratada por el servidor *Proxy.* Si la conexión es aceptada (ya que cumple con las reglas de *tn-gw*), tendremos que indicarle al servidor *Proxy*, cuál es la dirección a la que deseamos hacer la conexión *TELNET*.

FTP-GW

La instalación de un servidor *Proxy* para el servicio de *FTP* va a permitir trabajar de una forma parecida a la mostrada con el de *TELNET*, es decir, primero debemos realizar una conexión a la máquina que trabaja como cortafuegos, en la cual tenemos instalado el servidor *Proxy* de *FTP* y desde aquí realizar una conexión *FTP* al exterior, pero utilizando el servidor *Proxy*. Por este motivo no necesitamos, ni para *TELNET* ni para *FTP*, que en el cortafuegos existan cuentas para los usuarios, ya que en el caso de que se necesite algún tipo de autentificación, para aumentar la seguridad, *Fwtk* pone a nuestra disposición un servidor de autentificación propio. De esta forma evitamos el mayor problema de seguridad, las *password* utilizadas por los usuarios en sus cuentas.

Igual que hemos hecho para *tn-gw*, para el *Proxy FTP* también tendremos que modificar el fichero **inetd.conf**, para que se ajuste a nuestras nuevas necesidades. La línea donde se indica cómo se deben tratar las conexiones *FTP* se deberá sustituir por esta otra.

```
ftp stream tcp nowait root /usr/local/bin/ftp-gw ftp-gw
```

En el listado 1 tenemos un ejemplo de cómo podría ser la configuración básica de un *Proxy FTP*. Como vemos, la construcción de las reglas es bastante parecida por no decir igual a la sección del *Proxy TELNET*. Podemos fijarnos en la regla:

ftp-gw: permit-hosts 111.112.* -dest 190.1.1.* -deny {stor list}

Aquí tenemos un ejemplo de cómo utilizar los parámetros **-dest** y **–deny.** Se permite la conexión de todas aquellas máquinas cuya IP pertenezca al dominio **111.112.***, cuyo destino sea alguna máquina del dominio **190.1.1.*** mientras que la operación que se va a realizar no sea de almacenamiento o de listado del directorio.

tn-gw es el servidor Proxy para TELNET que ofrece TIS con Fwtk

Vamos a ver cuáles son los pasos que un usuario de nuestra red tendría que realizar para acceder a un servidor *FTP* situado en el exterior, por ejemplo en Internet, **ftp.publico.com.**

En primer lugar el usuario debe realizar una conexión *FTP* al cortafuegos mediante algún cliente *FTP*.

```
c:\ftp firewall.mired.com
```

Una vez que se haya conectado con el cortafuegos, éste le pedirá la

dirección del servidor *FTP* destino, así como el nombre de usuario que se utilizará para entrar en dicho servidor, con la siguiente nomenclatura **usuario@servidor.** Si quisiéramos conectarnos con el servidor **ftp.publico.com** mediante un usuario anónimo deberíamos utilizar la siguiente cadena:

```
anonymous@ftp.publico.com
```

En este paso, el cortafuegos aplicará las reglas de acceso y si las reglas permiten el acceso entonces realizará la conexión con el servidor *FTP* y se registrará con el *login* que el usuario ha facilitado (para nuestro ejemplo **anonymous**). Si el *login* con el que el usuario intenta conectarse necesita un *password*, el servidor *FTP* lo pedirá y en caso de ser correcto se habrá realizado la conexión con éxito.

HTTP-GW

Uno de los elementos más importante de los que ofrece la herramienta *Fwtk* es el servidor *Proxy* para acceso a la *Web*. Para configurar nuestro cortafuegos como un servidor *Proxy* de este tipo, debemos tener en cuenta que en este caso nuestro sistema se debe comportar de una forma parecida a como lo haría en el caso de que tuviéramos instalado un servidor *HTTP* en nuestra máquina.

No necesitamos, ni para TELNET ni para FTP, que en el cortafuegos existan cuentas para los usuarios

Debemos configurar el demonio **inetd** para que acepte peticiones por los puertos que normalmente se utilizan para este servicio. Los clientes *HTTP* realizan sus peticiones al puerto **80/tcp**, por lo que debemos añadir una línea como la siguiente en el fichero de servicios **/etc/services**.

```
httpd   80/tcp
```

Inetd ya conoce por qué puerto debe vigilar, por lo que ahora debemos realizar las modificaciones convenientes en el fichero de configuración **/etc/inetd.conf**, para que, como ocurría anteriormente con los servicios de *FTP* y *TELNET*, sea el servidor *Proxy http-gw* el encargado de tratar las conexiones con servidores en la red externa. Esta es la línea que debemos o bien modificar o bien incluir en el fichero **/etc/inetd.conf**.

```
httpd stream tcp nowait root /usr/sbin/
    http-gw http-gw
```

Como siempre que realizamos alguna modificación en la configuración de **inetd**, mandamos una señal *HUP* al demonio para que actualice su configuración.

```
kill -HUP pid_inetd
```

Como todos los servidores de *TIS*, *http-gw* cuenta con una sección de configuración en el fichero **/etc/netperm-table**, donde podremos especificar las reglas de acceso que el *Proxy* aplicará a las conexiones para este servicio, exactamente igual a como hemos visto con *TELNET* y *FTP*. Pero a diferencia de lo que ocurre con los dos anteriores que sólo tienen una función (el primero de conexión a un sistema remoto y el segundo para la transferencia de ficheros), el servicio *HTTP* lo podemos utilizar para acceder a páginas *web* o como cliente *FTP*. Por lo tanto, en las reglas podemos indicar el tipo de operación que se realiza, como por ejemplo:

```
http-gw : permit-hosts 190.100.*  -deny ftp
```

Con la línea anterior se permite el acceso a todas la máquinas cuya IP pertenezca a **190.100.***, pero se deniegan todas las operaciones de *FTP* utilizando *HTTP*.

Inetd ya conoce por qué puerto debe vigilar

Ahora nuestro sistema está preparado para trabajar como servidor *Proxy* de *HTTP*, por lo que tendremos que configurar los clientes *HTTP* para que conozcan cuál es la máquina encargada de trabajar como servidor *Proxy*. En todos los navegadores actuales se puede configurar esta característica indicando la dirección de la máquina que trabaja como *Proxy*, que en nuestro caso sería la dirección de la máquina que hace las labores de cortafuegos. Además debemos indicar el número del puerto por el que el servidor *Proxy* atiende las peticiones de servicio, tal y como se aprecia en la figura 1.

Tabla 2. Comandos de authsrv.

Adduser usuario	Añadir un usuario.
deluser usuario	Borrar un usuario.
display usuario	Información sobre un usuario.
enable usuario	Habilita/inhabilita a un usuario.
List	Lista todos los usuarios.
Password usuario	Modifica la contraseña del usuario.
Proto usuario protocolo	Establece el protocolo de autentificación para un usuario.
Quit	Salir.
Help	Ayuda.

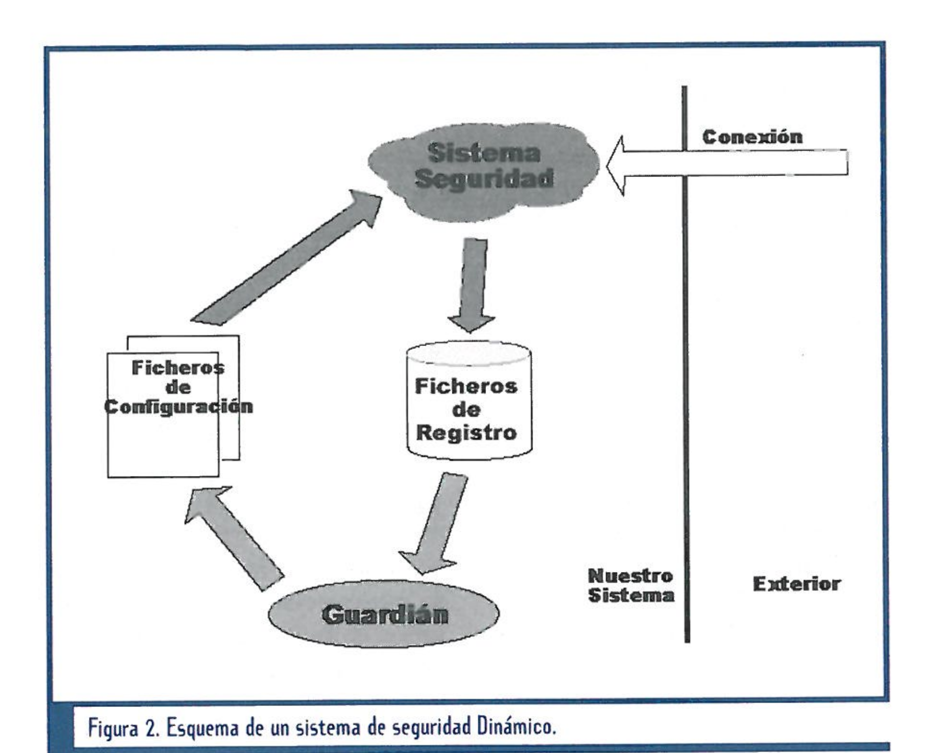


Figura 2. Esquema de un sistema de seguridad Dinámico.

SERVIDOR DE AUTENTIFICACIÓN

Como ya comentamos en el artículo anterior, los sistemas de autentificación están teniendo un enorme auge actualmente, por ser (no nos cansaremos de repetirlo) las *passwords* usadas por los usuarios el talón de Aquiles de la seguridad informática. Podemos encontrar sistemas de autentificación tan populares como *Kerberos*, sistemas como *S/KEY* u *OPIE* los cuales, recordemos, trabajan con *passwords* usadas una sola vez, protocolos como *SSL*, etc.

En todos los navegadores actuales se puede configurar la dirección de la máquina que trabaja como Proxy

Y como no podía ser de otra forma, TIS ha añadido el servidor de autentificación **authsrv** dentro de *Fwtk*. Este servidor permite tener un sistema de autentificación extra para cada uno de los servicios suministrados por el cortafuegos. Como ejemplo, en el listado 1, en el que aparece cómo podría ser la configuración de un *Proxy TELNET*, podemos ver que la regla utiliza una opción de acceso **-auth**, lo que indica debe solicitar autentificación.

Fwtk permite crear un sistema de autentificación distribuido, con una serie de servidores de autentificación repartidos por nuestra red o bien situados en los distintos cortafuegos de los que conste nuestro esquema de seguridad. Cuando deseemos que alguno de los servicios que ofrece *Fwtk* utilice un servidor de autentificación, debemos añadir una línea como la siguiente. En esta ocasión lo veremos para el caso del servidor *Proxy TELNET*.

```
tn-gw: authserver  HOST  PUERTO
```

Donde *HOST* sería el nombre de la máquina que tiene el servidor de autentificación y *PUERTO* el puerto hacia el que se debe dirigir la solicitud.

Para instalar el servidor de autentificación, debemos asignarle un puerto y añadir una línea en **/etc/inetd.conf**, para tratar las peticiones a dicho servicio:

```
authsrv stream tcp nowait roo
    /usr/local/bin/authsrv authsrv
```

Authsrv, como todos los servidores que ofrece *TIS*, tiene una sección de configuración en el fichero **/etc/netperm-table**, la cual es bastante parecida a las secciones de *tn-gw*, *tn-ftp*, etc. En ella configuraremos el lugar donde se encuentra la base de datos de los usuarios, tiempos de espera, mensaje de aviso, etc.

Además de la sección de configuración, **authsrv** consta de una base de datos donde se encuentra la información sobre los usuarios que necesitan autentificación. La administración de esta base de datos se realiza ejecutando **authsrv** desde la línea de comandos, alguna de las acciones que podemos efectuar se encuentran listadas en la tabla 2.

Authsrv consta de una base de datos donde se encuentra la información sobre los usuarios que necesitan autentificación

Para finalizar, ya hemos comentado que podemos tener varios servidores de autentificación, para administrar estos servidores remotamente, necesitamos una *shell* segura. No tiene sentido disponer de varios servidores de autentificación cuando la administración remota de los mismos se realiza mediante *TELNET*. *TIS* proporciona una *shell*, **authmgr**, totalmente segura con la que podremos realizar todas las tareas de administración de los distintos servidores de autentificación.

SISTEMAS DINÁMICOS VS ESTÁTICOS

Como hemos visto, la herramienta *fwtk* de *TIS* incorpora un *TCP Wrapper* propio, el cual recibe el nombre de *netacl*.

Tanto *TCP Wrapper* como *netacl* permiten definir una serie de reglas para evitar que desde una serie de máquinas o desde dominios determinados se puedan acceder a nuestro sistema, pero ¿qué ocurre si se intenta un ataque a nuestro sistema desde una máquina cuya IP no está registrada en ninguna de las reglas?

En este caso tendremos que esperar a que detectemos el ataque y una vez conocida la fuente del mismo, añadir la IP a alguna de las reglas. Aunque este sería el procedimiento normal, ¿cuánto tiempo se necesita para detectar un ataque? No todo el mundo puede estar las 24 horas del día delante de la consola revisando los ficheros de auditoría.

Supongamos el siguiente escenario, es la 1 de la mañana, nuestro sistema debe estar conectado las 24 horas del día a Internet, hemos configurado como mínimo alguna herramienta para filtrar las conexiones, como *TCP Wrapper* o *netacl* y alguien en alguna parte intenta acceder a nuestro sistema con intenciones poco correctas. Este es el primer ataque que se realiza desde esa máquina, con lo cual su IP no está registrada en nuestras reglas de filtrado, por lo que el atacante tiene por delante 7 horas de intentos, hasta que nosotros lleguemos y comprobemos los ficheros de auditoría. Esto resulta demasiado tiempo para aguantar un ataque y sobre todo si dicho ataque está bien organizado, así que se presenta un nuevo problema relacionado con todos aquellos sistemas de seguridad cuyo esquema es totalmente estático. Es decir, frente a un ataque se responde según unos ficheros de configuración, pero el sistema mismo no tiene capacidad de analizar el ataque y decidir en consecuencia una serie de acciones para acabar con dicho ataque.

Una idea en la que estamos trabajando consiste, en un software, el cual está continuamente analizando los ficheros de auditoria de nuestro sistema, para detectar posibles anomalías. Cuando encuentra algo sospechoso actúa según la acción que haya detectado, como por ejemplo añadiendo la IP del atacante a la lista de IP no permitidas, cerrando el servicio que está siendo atacado, etc.

No tiene sentido disponer de varios servidores de autentificación cuando la administración remota de los mismos se realiza mediante TELNET. TIS

De esta forma conseguimos crear un sistema dinámico, el cual puede detectar ataques en aquellos momentos en los cuales el administrador no se encuentra delante de la consola.

En la figura 2, podemos ver un esquema de un sistema dinámico. Consta de cuatro elementos, tres de ellos pertenecientes a un esquema tradicional y uno nuevo que se ha denominado **Guardián:**

- Alguna de las numerosas herramientas de filtrado, *TCP Wrapper*, *netacl*, etc.

- Los ficheros de registro del sistema.

- Los ficheros de configuración de las herramientas de filtrado.

- El proceso **Guardián**, encargado de analizar los ficheros de registro y modificar la configuración del sistema de seguridad.

El flujo de información se identifica con la dirección de las flechas.

CONCLUSIÓN

La experiencia nos dice que aunque instalemos un cortafuegos y dispongamos de la topología y configuración más adecuada que haga nuestro sistema totalmente seguro, éste nunca lo va a ser al 100%.

Siempre existirá alguna grieta y esto es debido a diversas causas, donde la mayoría aparecen relacionadas con el Administrador o usuario final: una mala configuración del cortafuegos al establecer las reglas poco estrictas, falta de experiencia del Administrador en los temas de seguridad, falta de tiempo para vigilar las posibles deficiencias del software instalado, de su configuración, así como la actualización del mismo, etc.

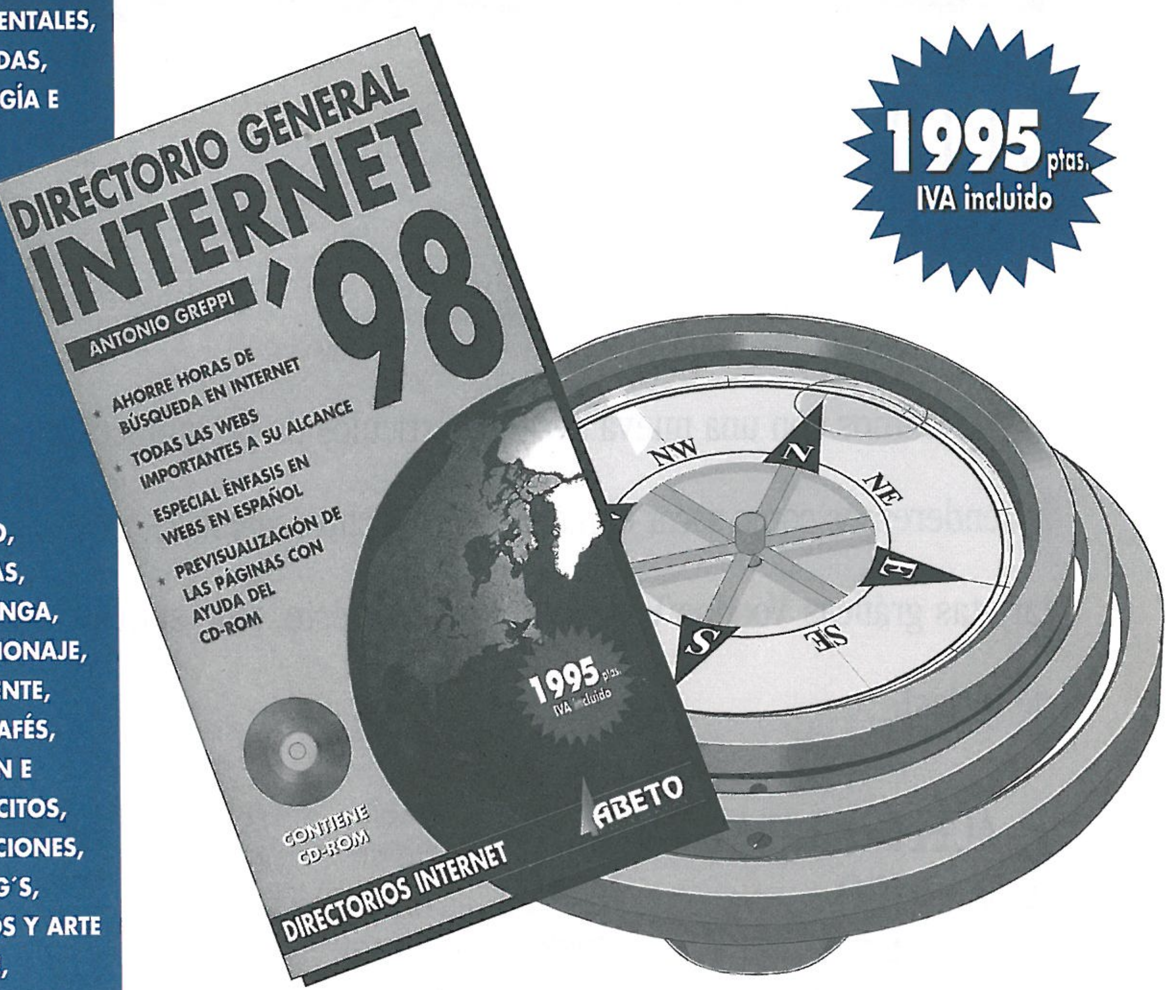
Otra causa muy importante es la amenaza de las fuentes internas, los usuarios que poseen cuenta en el sistema tienen acceso, aunque sea de lectura, a los ficheros del sistema y con la ayuda de programas tales como **hardcrack** (sunsite.unc.edu), **qcrack** (fpt.infospace.com/pub/qcrack), **rjgcrack** (sunsite.unc.edu) pueden llegar a romper el sistema.

Estas dificultades dirigen nuestros esfuerzos hacia nuevas investigaciones, con el fin de poder perfeccionar los sistemas de seguridad. En el apartado *Sistemas Dinámicos vs Estáticos*, hemos intentado avanzar en este sentido.

Estos son los 50 temas que contiene el libro:

AGENCIAS GUBERNAMENTALES, ALIMENTACIÓN Y BEBIDAS, ANIMALES, ARQUEOLOGÍA E HISTORIA, ARTE Y ARQUITECTURA, ASTRONOMÍA Y ESPACIO, AUTOMÓVILES, AVIACIÓN, BANCA Y FINANZAS, BARES, RESTAURANTES, DISCOTECAS, CASA Y DECORACIÓN, CINE, CIUDADES DEL MUNDO, COMERCIO Y EMPRESAS, CÓMICS, ANIME Y MANGA, CRIMEN, POLICIA, ESPIONAJE, CULTURAS, RAZAS, GENTE, REFUGIADOS, CYBERCAFÉS, DEPORTES, EDUCACIÓN E INVESTIGACIÓN, EJÉRCITOS, FOTOGRAFÍA, FUNDACIONES, ASOCIACIONES Y ONG´S, GEOGRAFÍA, GRÁFICOS Y ARTE GRÁFICO, HARDWARE, HOTELES, HUMANIDADES Y RELIGIÓN, HUMOR, INFORMACIÓN ÚTIL, INTERNETJUEGOS DE MESA Y JUGUETES, LIBROS Y EDITORIALES, MEDICINA Y SALUD, MEDIO AMBIENTE, MODA, JET SET Y COSMÉTICA, MOTOCICLISMO, MUSEOS, MÚSICA, NIÑOS Y ADOLESCENTES, PARQUES DE ATRACCIONES, PRENSAY REVISTAS, SEXUALIDAD, SOFTWARE TELECOMUNICACIONES Y ELECTRONICA, TELEVISIÓN Y RADIO, UNIVERSIDADES, VIAJES, TURISMO Y TRANSPORTES, VIDEOJUEGOS

¡NO TE LO PIERDAS!



Para orientarte en la Red, ahorrando tiempo y dinero.

Incluye CD-ROM con 2.500 direcciones y capturas de las páginas en color de las mismas.

Distribuidores autorizados

COMUNIDAD DE MADRID / CASTILLA LA MANCHA
DISTRIFORMA S.A.
Tfno. 91 - 501 4749

CATALUÑA
MIDAC LLIBRES S.L.
Tf. 93-421 18 95

ANDALUCIA OCCIDENTAL/ EXTREMADURA
DISTRIBUCIÓN DE EDICIONES RDGUEZ. SANTOS S.L.
Tfno. 95 - 418 04 75

ANDALUCIA ORIENTAL
DISTRIBUCIONES DEL MEDIODIA S.A. (ZÓCALO)
Tfno. 958-550278

CASTILLA - LEÓN
ARCADIA S.L.
Tfno. 983-395049

GALICIA
LUIS REY ABELLA (DISGALIBRO)
Tfno. 981-795754

ASTURIAS - CANTABRIA
DISTRIBUCIONES CIMADEVILLA S.A.
98-5167930

CANARIAS
GARCIA PRIETO LIBROS S.L.
Tfno. 922-820026

BALEARES
PONENT LLIBRES S.L.
971-430339

VALENCIA - CASTELLÓN
ADONAY S.L.
96-3975148

ALICANTE - MURCIA - ALBACETE
LA TIERRA LIBROS S.L.
Tfno. 96-5110192

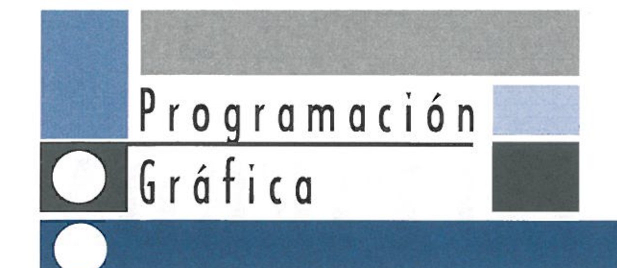
PAÍS VASCO - NAVARRA
YOAR S.L.
Tfno. 948-302239

ARAGÓN - LA RIOJA
ICARO DISTRIBUIDORA S.L.
Tfno. 976-126333

Ya a la venta en quioscos y librerías.



c/ Aragoneses, 7. 28.108 Alcobendas (MADRID)
Tel.: (91) 661 42 11* Fax: (91) 661 43 86



GLIDE (I)

Constantino Sánchez Ballesteros (constantino@nexo.es)

Comenzamos con una nueva serie de artículos de programación gráfica en los que aprenderemos como sacar el máximo provecho a nuestras, costosas pero potentes, tarjetas gráficas Voodoo/Voodoo 2. Por supuesto, todo será a golpe de render... ¡Y de teclas!.

ACELERADORAS VOODOO GRAPHICS

Todas las tarjetas gráficas actuales que se precien (y no me refiero al valor monetario) deben llevar chips *3DFX Voodoo/Voodoo 2* para obtener la mayor potencia y riqueza visual que se le puede pedir a un juego que represente gráficos 3D en tiempo real.

Voodoo Graphics es el primer subsistema de vídeo de bajo coste (bajo coste relativo) que permite aplicaciones 3D en tiempo real con las mejores prestaciones del mercado. Estas aceleradoras permiten aceleración de características 3D avanzadas tales como perspectiva real en mapeado de texturas, *mipmapping trilinear*, fuentes de luz, *anti-aliasing* (suavizado) de texturas y polígonos, etc. Además, implementa funciones comunes en gráficos 3D como sombreado *Gouraud* en triángulos, *buffers Z*, *Alpha Blending* (transparencias) , etc.

La librería *Glide* (la cual utilizaremos para programar nuestra *3DFX*) posee un *set* de funciones de renderizado a bajo nivel que actúan como una microcapa software para todos los chips que integran la tarjeta, destacando entre ellos, *Texelfx* y *Pixelfx*. Glide permite una fácil y eficiente implementación de librerías 3D, juegos, *drivers*, etc.

¿POR QUÉ GLIDE?

G*lide* busca dos propósitos principales:

- Releva al programador de trabajos a nivel hardware como mantenimiento de registros de la tarjeta, trabajo con constantes y desplazamientos de los registros.
- Crea una abstracción de todo el hardware *Voodoo Graphics* para facilitar el desarrollo a otras plataformas.

Resumiendo los detalles de las llamadas a bajo nivel que permite *Glide* a través de funciones de C/C++, los programadores podremos trabajar con registros específicos y la memoria de forma directa, permitiendo fáciles desarrollos y baja probabilidad de *bugs* a la hora de compilar y ejecutar.

Glide no implementa ninguna función que no acceda directamente al subsistema de la memoria o los registros

Glide es una parte del *kit* de desarrollo (*SDK*) de *3Dfx Interactive Software*, el cual se ha diseñado para permitir a todos los programadores



Figura 1. Logotipo de los chips 3DFX.

que puedan crear herramientas y juegos que estén optimizados para el hardware de *Voodoo Graphics*. *Glide* no es una *API* del tipo *OpenGL*, *Direct3D*, etc.; no permite operaciones gráficas 3D de alto nivel como transformaciones, manejo de listas, o sombreado de las fuentes de luz aplicadas en la escena. *Glide* sólo implementa aquellas operaciones que son nativamente soportadas por el hardware de la tarjeta. Por tanto, *Glide* no implementa ninguna función que no acceda directamente al subsistema de la memoria o a los registros.

Incrementando el número de Texelfx decrementa el número de pasadas requeridas para crear las diferentes técnicas de mapeado de texturas

La librería *Glide* puede ser enlazada a una aplicación con o sin *Debug*. La versión *Debug* tiene chequeo de errores y validación de parámetros, aunque puede bajar el rendimiento de la aplicación considerablemente. Por norma general, cuando se desarrolla una aplicación debe utilizarse la versión *Debug*, y cuando ya esté depurada, se enlazará la versión *Release* para obtener el máximo rendimiento del programa.

¿QUE NECESITAMOS PARA EMPEZAR A PROGRAMAR?

Existen dos tipos principales de Glide:

- Dos Glide
- Win32 Glide

Dado el poco uso actual delsistema operativo DOS para videojuegos que se presta en estos días, nos centraremos en la versión del *Glide* para Windows.

Para poder compilar un buen ejecutable de *Glide Win32*, necesitaremos Microsoft Visual C++ 4.0 o superior. Se puede intentar compilar en otro compilador, valga la redundancia, pero no sé si será factible puesto que no lo he intentado.

La otra cosa que necesitaremos será el *Glide SDK* de *3Dfx* (www.3dfx.com). Actualmente está la versión 3.0. Debemos asegurarnos de tener siempre la última versión de los drivers Glide para la tarjeta, pues de lo contrario, tendremos serios problemas a la hora de ejecutar los programas. Una cosa más, ¿He mencionado que es necesario tener instalada en nuestro ordenador una tarjeta *3Dfx Voodoo* o *Voodoo Rush*??

INSTALACIÓN DE LAS LIBRERÍAS EN EL ENTORNO VISUAL C++

Antes de hacer cualquier clase de programación, necesitaremos preparar el entorno de Visual C++ para incluir las librerías correctas. A continuación se muestran los pasos que seguir antes de empezar a programar:

- Ejecutar Visual C y crea un nuevo proyecto (*workspace*).
- Si posees una tarjeta *Voodoo*, elige **Consola** desde la lista de tipos de proyectos. En tarjetas *Voodoo Rush* no funcionará.
- Si tienes una *Voodoo Rush*, elige **Aplicación**. No utilices el asistente de aplicaciones o pronto sabrás el verdadero significado del dolor y el sufrimiento (las librerías *MFC*).
- Elige **Herramientas/Opciones** e introduce los *paths* a las librerías *glide* y los *Include* en la sección *directorios*. Un ejemplo podría ser: **[DRIVE]:\3dfx\glide\lib\win32** (para las librerías) **[DRIVE]:\3dfx\glide\src\sst1 \include** (para los include)

- Ahora elige **Crear/parámetros** y ve a la pestaña *C/C++*. Cambia el nivel de *warning* de **3** a **1**.

- En *Definiciones de Preprocesador*, después de **WIN32,_CONSOLE, (WIN32,_WINDOWS,** para *Voodoo Rush*) escribe **_MSC_**

- Cambia la categoría a **Generación de código** y cambia el procesador a **Pentium** (los usuarios de VC 5.0 necesitarán variar de __CDECL * a __STDCALL).

- En la pestaña *Link*, introduce **glide3x.lib**dentro de la lista de ficheros objeto de librerías.

IMPLEMENTACIONES DE LAS VOODOO

Las *Voodoo* se sitúan en las ranuras PCI de nuestro ordenador (hasta que salga la versión *AGP*). La configuración principal del hardware lo forman dos chips principales (*Texelfx* y *Pixelfx*) y la memoria. Incrementando el número de *Texelfx* decrementa el número de pasadas requeridas para crear las diferentes técnicas de mapeado de texturas.

La librería Glide puede ser enlazada a una aplicación con o sin Debug

Los ordenadores con varias tarjetas "pinchadas" en el ordenador utilizan el modo *SLI*, obteniendo el doble de rendimiento en las operaciones de *render* puesto que utilizan el doble de *Texelfx*.

Glide y *Voodoo Graphics* soportan un amplio abanico de técnicas de *render*, incluyendo:

- Sombreado *Gouraud*: el programador asigna valores de rojo, verde, azul y *alpha* para cada vértice. *Glide* calculará automáticamente los gradientes asociados y el hardware pondrá el resultado final en el triángulo definido.

Cada Voodoo Graphics tiene su propia versión de variables para el estado de Glide, incluyendo un valor de color constante

- *Texture mapping* (mapeado de texturas): el programador asigna valores iniciales de texturas (s/w, t/w, 1/w) para cada vértice y *Glide* computa los gradientes. El hardware creará la iteración correspondiente y la corrección de perspectiva para el mapeado de texturas sobre cada triángulo.

- *Texture mapping* con fuentes de luz: se trata de una combinación de la técnica anterior y el sombreado *Gouraud* para visualizar las fuentes de luz reflejadas en los polígonos texturados.

- Descompresión de texturas: utiliza un esquema *YAB* de compresión que mapea valores de 24-bit *RGB* a 8-bit *YAB* con una pequeña pérdida en la precisión.

- *Depth buffering* (*buffer* de profundidad): *Voodoo Graphics* soporta renderizado de *depth buffers* sin pérdida de rendimiento en la aplicación. Esta característica es implementada en la memoria designada como *frame buffer*.

- *Pixel blending* (suavizado de *pixels*): El hardware soporta funciones de *alpha blending* para combinar los *pixels* que se pongan encima de otros obteniendo otro *pixel* de diferente color.

- *Fog* (niebla): soporta tablas de 64 entradas para efectos atmosféricos tales como niebla. Cuando está activada, se utiliza una representación en coma flotante de 14 bits para indexar las 64 entradas anteriormente mencionadas e interpolar los valores. El valor resultante de la tabla representará el nivel de mezcla y transparencia que se utilizará para representar el efecto.

- *Chroma-keying*. Utilizada para determinados efectos de transparencia en objetos.

- *Color dithering*: Fundido de colores para el sombreado de objetos mediante operaciones de 24 bits.

Hay que saber que las *Voodoo* y *Voodoo 2* sólo soportan *render* en modo pantalla completa y paletas de color a 16 bits (dependiendo de la memoria instalada), por lo tanto, en modo ventana no tendremos aceleración hardware. La nueva hornada de aceleradoras *Voodoo* (*Banshee*) permite aceleración 3D tanto en modo ventana como en *Fullscreen*.

INTRODUCCIÓN A GLIDE

Glide provee diferentes funciones para inicializar, detectar y configurar el hardware de *Voodoo*. Dos funciones, *grSstQueryHardware()* y *grSstQuery Boards()* sirven para detectar la presencia de las tarjetas. *grGlideInit()*, *grSstSelect()*, y *grSstWinOpen()* sirven parßa inicializar *Glide* y el hardware, deben ser llamadas en el orden citado antes de uti-

Tabla 1. Especificando un handle de ventana en grSstWinOpen().

La interpretación del argumento win depende del entorno del sistema, tal y como se muestra a continuación:

Entorno de sistema	Valor win
DOS	NULL
Win32, pantalla completa	NULL or win
Win32, consola	NULL
Win32 Glide aplicación	NULL or win

lizar cualquier otra función (excepto *grSstQueryHardware()* y *grSstQuery Boards()*). Si no se siguen estas pautas, incurriremos en una inestabilidad del sistema y puede que nos toque reiniciar el ordenador.

```
typedef struct
{
int num_sst;
        GrSstConfig_t SSTs[MAX_NUM_SST];
} GrHwConfiguration;

FxBool grSstQueryBoards( GrHwConfiguration
        *hwConfig )

FxBool grSstQueryHardware(
        GrHwConfiguration *hwConfig )
```

La nueva hornada de aceleradoras Voodoo (Banshee) ya permiten aceleración 3D tanto en modo ventana como en Fullscreen.

GrSstQueryBoards() determina el número de *Voodoos* instaladas y guarda este resultado en **hwConfig Ænum sst.** Ninguna otra información es guardada en la estructura en ese momento; *grSstQuery Hardware()* puede ser llamada después de *grGlideInit()* para obtener datos en el resto de la estructura. *grSst QueryBoards()* es la única función de *Glide* que puede ser llamada antes que *grGlideInit()*; no cambia el estado del hardware ni hace ninguna operación de *render*.

GrSstQueryHardware() detecta la presencia de una o más *Voodoo Graphics* y determina cómo están configuradas. Esta función debería ser llamada inmediatamente después de *grGlideInit()* pero antes que cualquier otra función.

GrSstQueryHardware() devuelve un valor booleano: **FXTRUE** indica que al menos se ha encontrado una *Voodoo*. El argumento *hwConfig* es un puntero a una estructura que será rellenada con información sobre el número y configuraciones de las aceleradoras encontradas.

La primera función de inicialización, *grGlideInit()*, activa la librería *Glide* y debe ser llamada antes de ejecutar cualquier otra función. Asigna memoria, establece punteros, e inicializa variables y contadores de la librería. No utiliza argumentos y no devuelve ningún valor.

```
void grGlideInit( void )
```

La siguiente función llamada para inicializar el sistema es *grSstSelect()*, que establece como actual el subsistema gráfico instalado en el ordenador. Debe ser llamada después de *grSstQueryHardware()* y *grGlideInit()* pero antes que *grSstWinOpen()*.

```
void grSstSelect( int whichSST )
```

Glide es una API de bajo nivel, lo que nos obliga a crear nuestro propio engine de render desde un principio

El argumento representa el número ordinal del subsistema gráfico que será activo y debe estar dentro del rango *[0...*hwconfig.num_sst*]*, donde *hwConfig* es la estructura que alberga la información de la configuración del sistema devuelta por la llamada precedente a la función *grSstQuery Hardware()*. Si *whichSST* se establece

Tabla 2. Formatos de color.

Formato de color	Orden del byte			
GR_COLORFORMAT_RGBA	red	green	blue	alpha
	31 24	23 16	15 8	7 0
GR_COLORFORMAT_ARGB	alpha	red	green	blue
	31 24	23 16	15 8	7 0
GR_COLORFORMAT_BGRA	blue	green	red	alpha
	31 24	23 1	15 8	7 0
GR_COLORFORMAT_ABGR	alpha	blue	green	red
	31 24	23 1	15 8	7 0

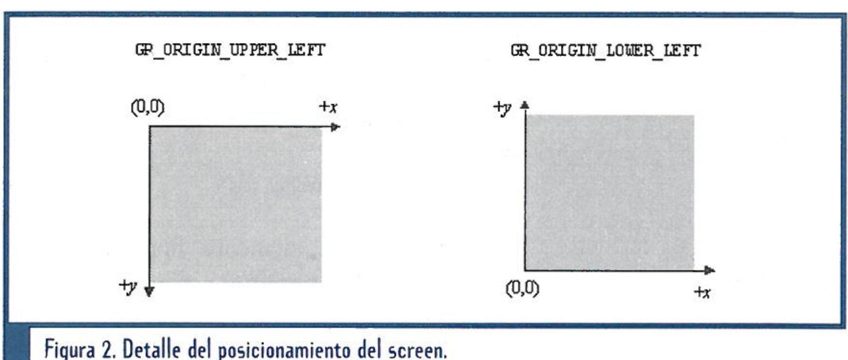


Figura 2. Detalle del posicionamiento del screen.

fuera del rango de valores y se utiliza la versión *Debug* de *Glide*, obtendremos un error en tiempo de ejecución.

La función final de inicialización, *grSstWinOpen()*, inicializa la *Voodoo* actual especificada por *grSstSelect()*. En este punto, todos los efectos especiales por hardware (*depth buffering*, *fog*, *chroma-key*, *alpha blending*, *alpha testing*, etc.) están desactivados. Todas las constantes globales de estado (valor de referencia de *chroma-key*, valor de referencia de *alpha*, valor de profundidad, etc.) y contadores se inicializarán con el valor **0**.

Glide provee diferentes funciones para inicializar, detectar y configurar el hardware Voodoo.

GrSstWinOpen() debería ser llamada una vez por cada *Voodoo Graphics* instalada (cuando se utiliza el modo *SLI*, dos Voodoo juntas, se tratan como una sola tarjeta) y debe ser ejecutada después de *grGlideInit()*, *grSstQuery Hardware()* y *grSstSelect()*. Esta función retorna **FXTRUE** si la inicialización fue un éxito o **FXFALSE** en caso contrario.

```
FxBool grSstWinOpen( FxU32 hwnd,
       GrScreenResolution_t  res,
       GrScreenRefresh_t     refresh,
       GrColorFormat_t       cFormat,
       GrOriginLocation_t    locateOrigin,
       int                   numBuffers,
       int                   numAuxBuffers
       )
```

Los argumentos de *grSstWin Open()* configuran el *frame buffer*. El primero de ellos, *win*, especifica un *handle* para la ventana en la que se visualizarán los gráficos.

La interpretación de *win* depende del entorno del sistema, de tal forma que los programas para MS-DOS deben especificar el valor **NULL**, mientras que las aplicaciones sobre *Win32* a pantalla completa deben especificar un *handle* de una ventana.

La resolución y el refresco de pantalla son especificados en los dos argumentos siguientes, *res* y *refresh*. Ambas variables representan valores elegidos de los tipos definidos en el archivo header **sst1vid.h**. Una aplicación podría establecer *res* como **GR_RESOLUTION_640x480** y *refresh* como **GR_REFRESH_60HZ.** El parámetro *ref* es ignorado cuando una aplicación *Win32* se ejecuta en modo ventana.

El cuarto argumento, *cFormat*, especifica el orden de color *RGBA* del *frame buffer*. Diferentes sistemas software asumen diferentes formatos para los datos de color del *pixel*. En la siguiente tabla se muestran los diferentes formatos de color que podemos utilizar:

El quinto parámetro en *grSstWin Open()* especifica la localización del origen de la pantalla. Si *locateOrigin* es **GR_ORIGIN _UPPER_LEFT**, la pantalla se situará en la esquina superior izquierda, mientras que **GR_ORIGIN_ LOWER_LEFT** colocará el origen de la pantalla en la esquina inferior izquierda.

Los dos argumentos finales de *grSstWinOpen()* seleccionan las opciones de *buffering*. La primera, *numBuffers*, especifica doble o triple *buffer* y es un valor entero (2 ó 3). El otro argumento, *numAuxBuffers*, especifica el número de *buffers* auxiliares requeridos por una aplicación, los cuales son utilizados para *depth* o *alpha buffering*. Los valores permitidos son 0 y 1.

Los argumentos de grSstWinOpen() configuran el frame buffer

Si no hay suficiente memoria para soportar la resolución deseada y las opciones de *buffer* obtendremos un error. En el siguiente fragmento de código veremos la secuencia típica para la inicialización de *Glide*. Este código llama a las cuatro funciones comentadas anteriormente en el orden requerido. Inicializarán el software y el hardware.

Los parámetros de *grSstWin Open()* establecen un doble buffer a pantalla completa, **640¥480** de resolución y **60Hz** de refresco de pantalla. Los colores son guardamos como *RGBA* (*alpha RGB*), el origen se sitúa en la esquina inferior izquierda y no existe *buffer* auxiliar.

```
GrHwConfiguration hwconfig;

grGlideInit(void);

if (grSstQueryHardware(&hwconfig))
{
       grSstSelect(0);
       grSstWinOpen(  NULL, GR_RESOLU-
```

```
        TION_640x480, GR_REFRESH_60HZ,
        GR_COLORFORMAT_RGBA, GR_ORI-
        GIN_LOWER_LEFT, 2, 0);
};
else printf("ERROR: no Voodoo Graphics!\n");
```

Glide soporta dos formas de instalación de las *Voodoo*:

- Múltiples *Voodoo Graphics* creando múltiples visualizaciones.
- Dos *Voodoo Graphics* utilizando de forma conjunta una sola visualización.

SELECCIONANDO UNIDADES VOODOO

En este momento del código, sólo tenemos un sistema *Voodoo* activo.

GrSstSelect() activa una unidad específica. Todas las funciones *Glide*, con la excepción de la familia **grGlide** y **grSstSelect()**, operan sólo sobre la *Voodoo* actual que se encuentre activa.

Glide permite utilizar hasta Triple Buffering

Hay que hacer constar que el estado global de *Glide* se refiere a cada *Voodoo* de forma independiente. Por ejemplo, para asignar el color constante en cada unidad *Voodoo Graphics* al mismo valor debemos escribir un bucle que seleccione cada una y asigne el color, tal y como se muestra en el siguiente extracto de código. Cada *Voodoo Graphics* tiene su propia versión de variables para el estado de *Glide*, incluyendo un valor de color constante que será utilizado para borrar la pantalla. El color constante es cero, por defecto.

En el siguiente código mostrado a continuación se establece un bucle a través de todas las unidades *Voodoo* encontradas por la llamada de la función *grSstQueryHardware()*, y asignará el color constante a negro:

```
GrHwConfiguration hwconfig;

for ( I = 0; I < hwconfig.num_sst; I++ )
{
        grSstSelect( I );
        grConstantColorValue( ~0 ); /* only
        affects SST "I" */
}
```

La función final de inicialización, grSstWinOpen(), inicializa la Voodoo actual especificada por grSstSelect().

GrSstWinOpen() debe ser llamado cada vez que se utilice cada *Voodoo Graphics*. Conviene recordar que dos *Voodoo* unidas mediante *SLI* serán tratadas por software como una única tarjeta, aunque el rendimiento obtenido será del doble.

MODO SLI

Como ya se ha dicho, dos *Voodoo Graphics* pueden unirse mediante un cable especial en una configuración especial llamada *SLI* (*scanline interleaving*). Desde la perspectiva de una aplicación, esta configuración es tratada tratadas como una simple tarjeta instalada en el ordenador, incluyendo cuando se efectúa la detección del hardware instalado, inicialización, descarga de texturas, etc.

Después de que una aplicación ha completado la utilización de *Glide* y *Voodoo*, es necesario terminar la aplicación de forma correcta. En el siguiente apartado se mostrará este proceso con mayor detalle.

FINALIZANDO LA APLICACIÓN

La función *grGlideShutdown()* desactiva Glide y todas las tarjetas previamente abiertas mediante *grSstWin Open()*. *grGlideShutdown()* debería ser llamada sólo cuando la aplicación ha terminado de utilizar *Glide*, y no debería ser ejecutada antes que las funciones *grGlideInit()* y *grSstWin Open()*.

```
void grGlideShutdown( void )
```

En el siguiente listado se visualiza una mínima aplicación en la que se muestra la forma de incializar y finalizar *Glide*:

```
#include <glide.h>
GrHwConfiguration hw;
void main(void)
{
grGlideInit(void);
if (! GrSstQueryHardware(&hw))
    printf("ERROR: no Voodoo Graphics!\n");
grSstSelect(0);
grSstWinOpen(NULL,
        GR_RESOLUTION_640x480,
        GR_REFRESH_60HZ,
     GR_COLORFORMAT_RGBA GR_ORI-
        GIN_LOWER_LEFT, 2, 0);
grGlideShutdown();
}
```

DISPLAY BUFFERS

Glide maneja varios *buffers* gráficos hardware, todos los cuales se basan en compartir el mismo área de memoria conocida como **frame buffer**. Depen diendo de la cantidad de memoria

instalada en la tarjeta, el *frame buffer* es utilizado por tres unidades lógicas:

- *front buffer*
- *back buffer*
- *buffer* auxiliar

void grRenderBuffer(GrBuffer_t buffer)

La función *grRenderBuffer()* selecciona el *buffer* para el dibujado de primitivas y el borrado del mismo. Los valores pueden ser: **GR_BUFFER_FRONTBUFFER** y **GR_BUFFER_BACKBUFFER**, siendo el último el que se utiliza por defecto.

El *buffer* auxiliar en la *Voodoo* se puede utilizar como *depth buffer* (*buffer* de profundidad), *alpha buffer*, o tercer *buffer* de renderizado para el *triple buffering*. El auxiliar no es accesible en sistemas con 2MB de *frame buffer* con *DRAM* corriendo a una resolución de 800x600. De todos modos, en sistemas con 4MB de *frame buffer DRAM* y 640x480 es viable su obtención.

Esta librería nos permite aprovechar al máximo las tarjetas basadas en 3DFX

El *triple buffering* permite a una aplicación continuar renderizando aún cuando un *buffer* de intercambio (*swap buffer*) está pendiente de realizar su trabajo. Una aplicación selecciona un *buffer* auxiliar cuando estén activados *depth buffering, alpha buffering o triple.*

Por ejemplo, si *grDepthBuffer Mode()* es llamado con un parámetro como **GR_DEPTHBUFFER_DISABLE** se asume que el auxiliar será utilizado para *depth buffering*. De forma similar *grSstWinOpen()* activa el *triple buffering*; *alpha buffering* es activado si *grAlphaBlendFunction()* selecciona un factor de destino del tipo *alpha blending* o *grColorMask()* permite la escritura al *buffer alpha*.

OCULTANDO EL FRAME BUFFER

La escritura al *frame buffer* puede ser activada o desactivada de forma selectiva. Las funciones *grColorMask()* y *grDepthMask()* controlan la ocultación del *buffer*:

- **FXTRUE**: permite escribir al *buffer* asociado
- **FXFALSE**: desactiva la escritura al *buffer* asociado

La escritura del color y el buffer alpha es controlada por grColorMask()

La escritura del color y el *buffer alpha* es controlada por *grColorMask()* mientras que la escirtura al *depth buffer* es controlada por *grDepthMask()*. Desactivar la escritura a los planos *alpha* es lo mismo que desactivar la escritura a los planos de profundidad dado que ambos comparten la misma memoria.

void grColorMask(FxBool rgb, FxBool alpha)

void grDepthMask(FxBool enable)

GrColorMask() especifica cuándo el color y/0 *buffers alpha* pueden o no ser escritos durante las operaciones de *render*. Si *rgb* es **FXFALSE**, por ejemplo, no se hará ningún cambio al color del *buffer* mientras se efectúe la operación de dibujo. El parámetro *alpha* se ignorará si el *depth buffering* está activado.

grDepthMask() activa la escritura al depth buffer.

VOLCADO DE BUFFERS

En un *buffer* de tipo doble o *triple buffer*, la siguiente escena será renderizada en el *backbuffer* mientras el *frontbuffer* se esté visualizando en la pantalla. Después de que la imagen se haya renderizado, se visualizará con una llamada a la función *grBufferSwap()*, la cual intercambia los datos gráficos del *backbuffer* y el *frontbuffer* después del valor *swapInterval* que corresponde al retrazo vertical. Si *swapInterval* es **0**, el volcado no esperará al retrazo vertical. Si la frecuencia del monitor es de **60 Hz**, por ejemplo, un *swapInterval* de **3** ofrecerá un *ratio* de *frames* máximo de **20 Hz**.

void grBufferSwap(int swapInterval)

Un *swapInterval* de **0** puede resultar pernicioso para el resultado final de la escena, dado que puede producirse un volcado de *buffers* durante el ciclo de refresco del monitor.

GrBufferSwap() no espera al retrazo vertical; dibuja en pantalla y retorna inmediatamente para dibujar la siguiente pantalla. Si la aplicación utiliza *doble buffering*, *Voodoo* parará el *render* y esperará hasta que ocurra el volcado de gráficos antes de ejecutar más comandos. Si la aplicación utiliza *triple buffering* y el tercer *render* es viable, los comandos de renderizado tendrán lugar de forma inmediata en el tercer *buffer* que conforma el *triple buffering*.

Una aplicación *Glide* puede utilizar la función *grBufferNumPending()* para determinar el número de *buffers* que están esperando para ser visualizados.

```
int grBufferNumPending( void )
```

GrBufferNumPending() retorna el número de *buffers* que están en cola. El valor máximo devuelto será **7**, aunque haya más en la cola. Para minimizar la latencia de *render*, *grBufferNum Pending()* debería llamarse en un bucle una vez por cada *frame* creado hasta que el valor devuelto se encuentre entre **1**, **2** ó **3**.

SINCRONIZANDO CON EL RETRAZO VERTICAL

La sincronización con el retrazo vertical es soportada mediante las funciones *grSstVRetraceOn()* y *grSstVideo Line()*. La primera devuelve **FXTRUE** si el monitor está dentro del retrazo vertical y **FXFALSE** si no lo está.

```
FxBool grSstVRetraceOn( void )
```

Voodoo y Voodoo 2 sólo soportan render en modo pantalla completa y paletas de color a 16 bits

GrSstVideoLine() retorna el número de línea actual que se está barriendo en la pantalla. Este número es **0** durante el retrazo vertical e incrementa mientras progresa el barrido de líneas hacia debajo del monitor. Existe un pequeño número de líneas que no son visualizadas en la parte superior de la pantalla. Para solventar este pequeño inconveniente, *grSstVideoLine()* retorna un número positivo cuando el barrido comienza al principio de la pantalla. De este modo, sabremos cuándo podemos comenzar a dibujar.

```
FxU32 grSstVideoLine( void )
```

BORRANDO BUFFERS

La habilidad de limpiar un *buffer* es fundamental para la animación de una escena, pues de lo contrario obtendríamos restos de frames anteriores en el actual. *Voodoo Graphics* permite que se borren el *back buffer* y *alpha* o *depth buffer* de forma simultánea. Una limpieza de *buffers* consiste en dibujar *pixels* de un mismo color antes de realizar sobre él las operaciones de renderizado.

Nota: La limpieza de buffers no es necesaria si se plasma continuamente una imagen de fondo en los buffers en lugar de pixels con el mismo color.

Los *buffers* son limpiados mediante la función *grBufferClear()*. El área del *buffer* que será limpiada vendrá delimitada por los valores asignados en *grClipWindow()*. Los tres parámetros especifican los valores que se utilizarán para limpiar el *backbuffer* (*color*), *alpha buffer* (*alpha*), y *depth buffer* (*depth*).

GrSstWinOpen() debería ser llamada una vez por cada Voodoo Graphics instalada

El parámetro *depth* puede ser una de las constantes **GR_ZDEPTHVALUE_NEAREST**, **GR_ZDEPTHVALUE_FARTHEST**, **GR_WDEPTHVALUE_NEAREST**, **GR_WDEPTHVALUE_FARTHEST**, o una representación directa de un valor en el *depth buffer*.

```
void grBufferClear( GrColor_t color, GrAlpha_t
        alpha, FxU16 depth )
```

Cualquier *buffer* activo es automática y simultáneamente borrado por *grBufferClear()*. Por ejemplo, si *depth buffering* está activado (*grDepthBuffer Mode()*), el *depth buffer* se borrará al valor de *depth*. Si *alpha buffering* está activado (*grAlpha BlendFunction()*), será borrado al valor de *alpha*. Si también está activada la escritura al *backbuffer* (*grColor Mask()*), se borrará al valor de *color*.

CONTROL DE ERRORES

Glide posee una familia de funciones para la gestión de errores cuando se depura la aplicación. Esta familia consiste en manejos de error de *Glide* y de la aplicación.

La versión Debug de Glide crea unos parámetros de validación y chequeo de errores

La versión *Debug* de *Glide* crea unos parámetros de validación y de chequeo de errores. Cuando se detecta una condición de error, se procederá a ejecutar una función *callback* creada por el programador. Esta función es instalada mediante la función *grErrorSet Callback()*. Si no se especifica ninguna función *callback*, se utilizará una función de error por defecto que mostrará un mensaje en la consola.

```
void grErrorSetCallback( void (*function)(const
        char *string, FxBool fatal) )
```

La función acepta una cadena de caracteres describiendo el error y un *flag* indicando si el error es fatal o recuperable. *grErrorSetCallback()* es relevante sólo cuando se utiliza la versión *Debug* de *Glide*; la versión *Release* borra todos los parámetros internos de validación y chequeo de errores, por lo que la función *callback* creada nunca será llamada.

NUESTO PRIMER PROGRAMA

A continuación vamos a ver el listado de nuestro primer programa en *Glide* haciendo algo específico: pintar la pantalla de rojo. No es que sea algo espectacular, pero es suficientemente ilustrativo para ir comprendiendo el uso de las funciones más comunes de esta librería. El proyecto se ha realizado para Visual C++ 5.0:

En primer lugar incluiremos en nuestro programa las librerías están-dar y **Glide.h**.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <glide.h>
```

Existen dos versiones de Glide: DOS y Windows. La elección dependerá del programador

El siguiente y único procedimiento (*main*) será el que albergue todas las instrucciones necesarias para realizar el propósito de nuestro programa. Es necesario crear una variable para albergar la configuración de nuestra tarjeta.

```
void main(void)
{
GrHwConfiguration HwConfig;
```

Inicializamos *Glide*. Esta función debe llamarse antes que cualquier otra para no incurrir en errores de algún tipo:

```
grGlideInit();
```

Comprobamos la presencia de hardware *3dfx*. Si se detecta hará que salga el logotipo de *3DFX* en la pantalla. En caso contrario, salimos del programa con un mensaje de error:

```
if ((grSstQueryHardware(&HwConfig)) ==
        FXFALSE)
  {
      printf("Error! No existe hardware 3dfx!");
      exit(1);
  }
```

Una vez encontrado lo que buscábamos, seleccionamos la primera *Voodoo* instalado en el sistema. Esto significa que podemos utilizar diferentes tarjetas de nuestro ordenador conectadas a diferentes monitores desde el mismo programa. *Glide* sólo trabajará con la seleccionada desde la función *grSstSelect*.

```
grSstSelect( 0 );
```

La habilidad de limpiar un buffer es fundamental para la animación de una escena

Establecemos el modo de vídeo a **800x600**, **60 Hz** de refresco de pantalla, formato de color **RGBA** y posición superior izquierda.

```
grSstWinOpen( 0,
        GR_RESOLUTION_800x600,
        GR_REFRESH_60Hz,
        GR_COLORFORMAT_RGBA,
        GR_ORIGIN_UPPER_LEFT,
        2, 1);
```

Ponemos el *Backbuffer* de color rojo (**0xff000000**):

```
grBufferClear( 0xff000000, 0, GR_WDEPTH-
      VALUE_FARTHEST );
```

Volcamos los datos gráficos desde el *backbuffer* al *frontBuffer*. Esto hará que visualicemos en pantalla el color rojo.

```
grBufferSwap(1);
```

El volcado de buffers nos permitirá visualizar en pantalla el resultado del render

Esperamos a que se pulse una tecla:

```
getch();
```

Acabamos con *Glide* y retornamos al modo de vídeo normal:

```
grGlideShutdown();

}
```

FUNCIONA, ¡QUE NO ES POCO...!

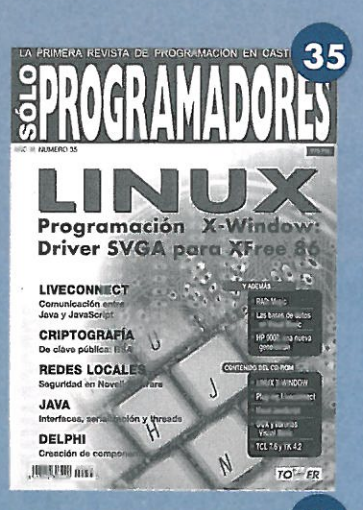
Ya veis que la programación en *Glide* no es tan complicada como pueda parecer en un principio. Tampoco es que hayamos logrado algo espectacular para la vista pero ¡funciona!. Y de eso se trata, de ir tomando contacto con la librería poco a poco.

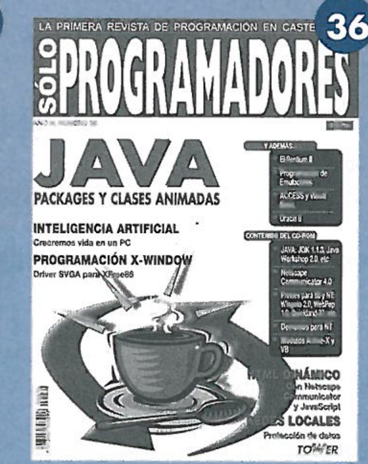
En la siguiente entrega podréis ver nuevas funciones y estupendos resultados visuales aplicados con nuestra excelente *Voodoo* y su inseparable *Glide*.

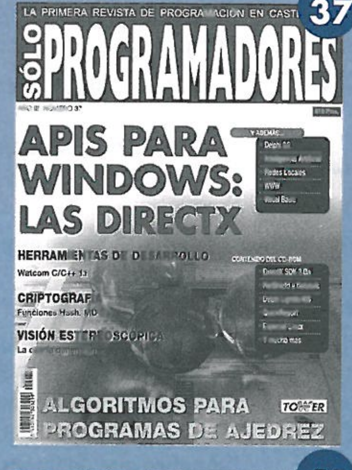
números atrasados

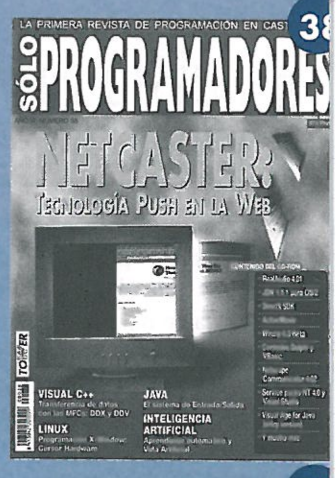
SÓLO PROGRAMADORES

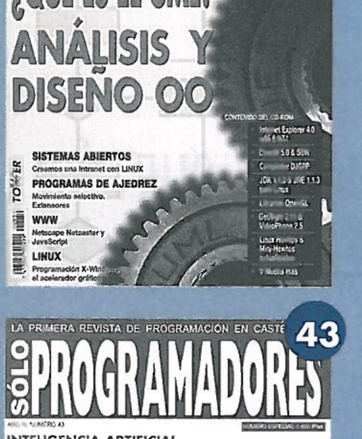
completa ya tu colección
























suscríbete

a Sólo Programadores

y consigue un **magnífico descuento**

suscripción
normal

ahorro **20%**

12 revistas
(1 año)
por sólo...

9.350 ptas.

suscripción
estudiantes
(carreras técnicas)

ahorro **40%**

12 revistas
(1 año)
por sólo...

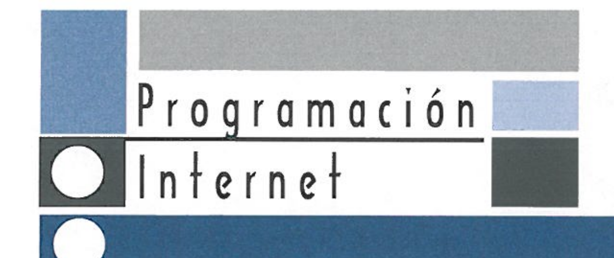
7.050 ptas.











Creación de un buscador Web (I)

Enrique de la Lastra (elastra@redestb.es)

Los buscadores Web permiten encontrar cualquier archivo o recurso que se encuentre en la red Internet y más concretamente en la Web. ¿Pero cómo se crea un buscador? A lo largo de esta nueva serie de artículos analizaremos todo el proceso de creación.

INTRODUCCIÓN

Desde los primeros tiempos de andadura la evolución de Internet estuvo acompañada de un problema en cada paso que se daba. Cada vez que se añadía un nuevo servidor con información se hacía más complicado localizar un determinado fichero o recurso. Hay que tener presente que la clave del éxito de Internet ha sido y sigue siendo la posibilidad de compartición de datos y la existencia de información y recursos distribuidos. Por tanto, aunque la comunidad internauta se beneficiaba de la inclusión de nuevos servidores, el acceso a un volumen de datos que iba en constante aumento, también crecía en dificultad de la misma manera.

Esta problemática llevó a idear mecanismos de búsqueda dentro de la maraña de ficheros en que se estaba convirtiendo Internet. Así se crearon los servicios *Archie*, *Gopher* o *WAIS*, que no son sino los antepasados de los actuales buscadores *Web*.

Archie era ("era", en pasado, no porque ya no exista, sino porque su funcionalidad ha sido sobrepasada con nuevas herramientas) un sistema de localización de ficheros a través de bases de datos que contenían la información almacenada en los servidores de *FTP* anónimo. Para utilizarlo era necesario un cliente que permitía conectarnos a los servidores *Archie* donde podíamos localizar un determinado archivo. En dicho cliente se introducía una cadena de búsqueda y en función de la misma se mostraban los servidores de *FTP* que contenían archivos relacionados con la misma. Si configurábamos un cliente *FTP* podíamos bajarnos directamente cualquiera de los archivos mostrados.

Gopher, a su vez, era (también en pasado, aunque siga existiendo) otro servicio en el que sus servidores organizaban la información en árboles jerárquicos sobre cuyas ramas íbamos descendiendo en función del tema elegido. De esta manera, al conectarnos a un servidor de este tipo, éste mostraba el árbol principal y al seleccionar una rama se conectaba a otro (o al mismo) servidor donde se encontraba el subárbol correspondiente. Cada vez que bajábamos una rama podíamos permanecer en el mismo servidor o pasar, de forma totalmente transparente para nosotros, a otro servidor. Al final se accedía al recurso deseado, que podíamos bajarnos directamente a través de *FTP*, o mediante una conexión mediante *Telnet* al servidor que lo contenía.

En estos primeros tiempos todavía no existían los servidores *Web*, ya que aún tardaría bastante en crearse el

lenguaje *HTML*, auténtico impulsor de la *Web*. Sin embargo, el servicio prestado por *Gopher*, aunque estaba concebido para ser distribuido, es muy similar a cualquier Índice Jerárquico de hoy día (por ejemplo, *Yahoo*, ver Figura 1), donde la información se estructura en temas y al ir seleccionando alguno de ellos se muestra una selección de subtemas y así sucesivamente hasta que llegamos (si es el caso) al recurso deseado.

Los Robots reciben varios nombres: "Bots", "Spiders", "Web crawlers", "Web wanderers", "Warms", etc.

El servicio de *Archie* era muy similar al de cualquier buscador *Web*, ya sea un Índice como *Yahoo* o un Motor de Búsqueda como *Lycos* (ver Figura 2).

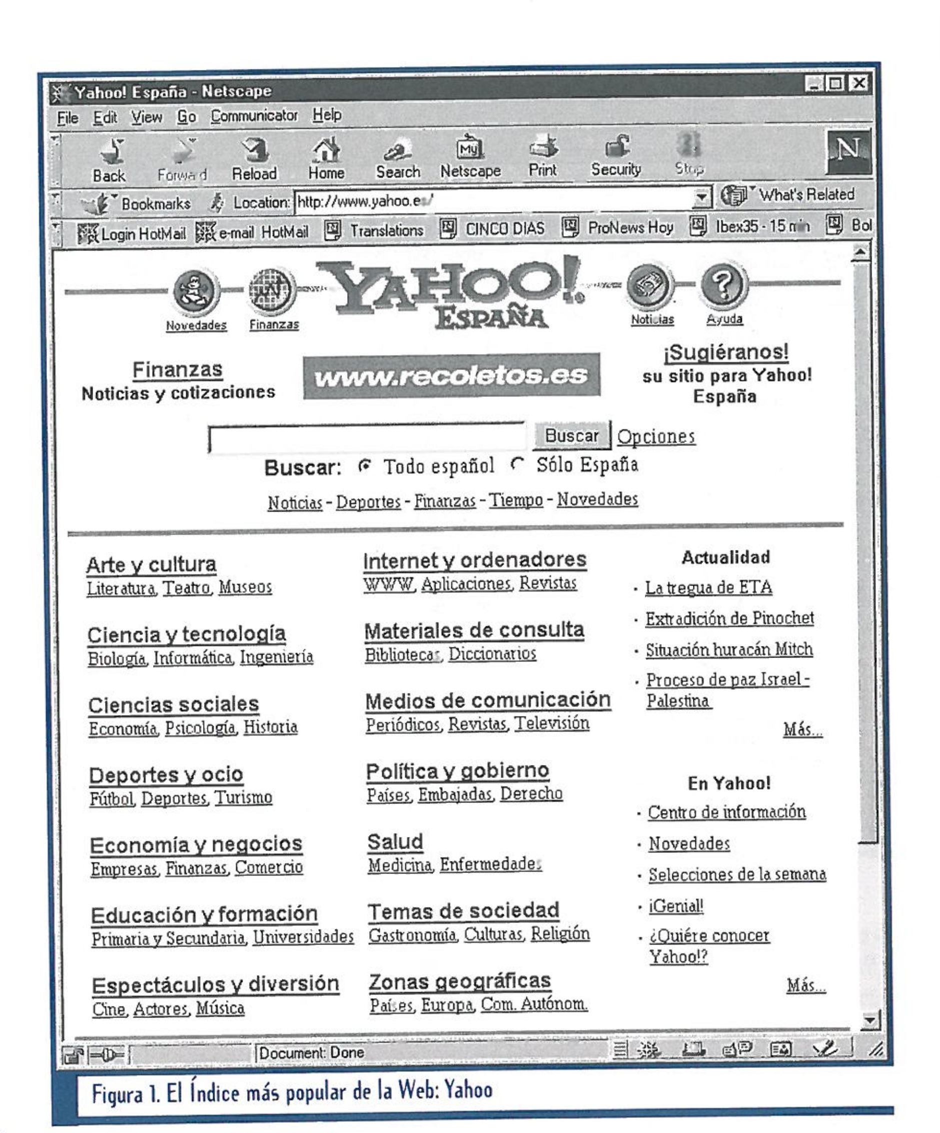


Figura 1. El Índice más popular de la Web: Yahoo

ROBOTS WEB

La diferencia básica entre un Motor de búsqueda y un Índice reside en la forma de conseguir la información para generar la base de datos sobre la que posteriormente el usuario realizará su búsqueda particular. Mientras que un Índice actualiza su información manualmente, a través de las solicitudes que le llegan de páginas que desean registrarse, un Motor de Búsqueda utiliza un *Robot* para actualizar y ampliar sus datos. Aunque los Índices son muy útiles cuando se tiene muy claro lo que se desea buscar o la información contenida cambia poco o nada, presentan una desventaja derivada de su actualización manual. Esta carencia consiste en que no contienen los últimos recursos que se publican en la Red asociados a un determinado tema.

Los Motores de Búsqueda, sin embargo, utilizan unos programas denominados *Robots*, también conocidos como *Spiders* (arañas), *Web crawlers*, *Web Wanderers* (deambuladores *Web*) o *Warms* (gusanos), cuya función básica es la de conectarse a servidores *Web* y recuperar información sobre las páginas a través de las cuales van navegando. El hecho de utilizar denominaciones como "gusanos" o "arañas" parece dar a entender que se mueven dentro de los servidores como si de virus se tratasen. Nada más lejos de la realidad: lo único que realizan son peticiones de páginas y después de analizar los enlaces y recursos referenciados, lanzan de forma automática nuevas peticiones de ficheros al mismo o a distintos servidores *Web*.

El nombre de *Robot* o *Bot* (esta última palabra no es sino una abreviatura), se debe a una obra de *Karel Capek's*, donde se utiliza el término **robota** que en checo significa trabajo.

Los Robots son programas que recorren e indexan las páginas HTML de forma automática

Veamos detenidamente en qué consisten estos "husmeadores". Los *Robots Web* son programas que recorren páginas de la *World Wide Web* mediante los enlaces que incluyen cada una de las páginas por las que van pasando.

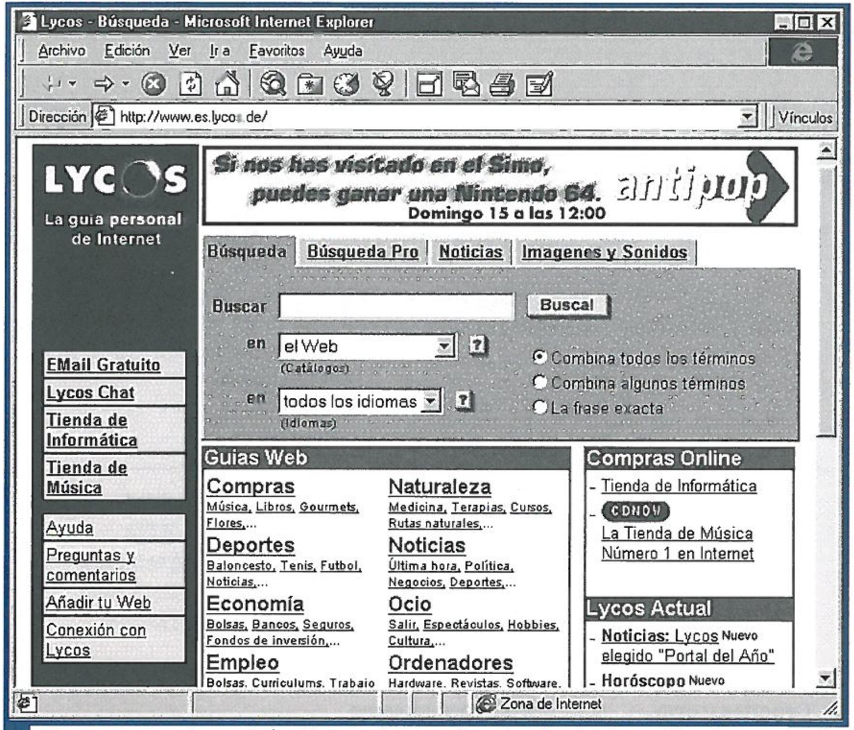


Figura 2.Imagen de uno de los Motores de búsqueda más utilizados: Lycos. Hoy día no es posible distinguir, por su apariencia, un Motor de búsqueda de un Índice.

El mecanismo de funcionamiento es el siguiente: se le pasa una dirección *URL* al *Robot*, éste se conecta al *Web* correspondiente, de igual manera a como lo puede realizar un navegador y solicita el recurso correspondiente (en un principio, debería ser una página *Web*). En la página que devuelve el servidor el *Robot* analiza unas etiquetas especiales en las que se le informa si puede indexar esa página (es decir, añadir la referencia a esa página en su Base de Datos) y si puede inspeccionar los enlaces que contiene la misma. En este último caso, solicita automáticamente todas las páginas *HTML* a las que esta página inicial contiene un enlace (y además puede tratar de indexar otros recursos referenciados en ella).

El comportamiento es, por tanto, similar al de un navegador, con la diferencia de que realiza la inspección de las páginas de forma automática. Además no tiene por qué almacenar las páginas que va descargando, sino que se puede limitar a analizar y almacenar la parte de las mismas que le interese.

UTILIDADES DE LOS ROBOTS

Visto lo anterior, deducimos de forma inmediata que las posibilidades de utilización de un *Robot* son muchas y variadas, y que de hecho su función asociada a un Motor de Búsqueda *Web* es sólo una de ellas.

Los Robots no sólo permiten crear Motores de Búsqueda, también son el origen de los Offline Browsers

Algunas utilidades de los *Robots*:

- **1.Creación de Offline Browsers** (navegadores sin conexión): es decir programas que se conectan a un determinado *URL* y se bajan todos los archivos que encuentran dentro del mismo. Los hay que hasta replican en el disco duro la estructura completa del *Web*, con el nombre del servidor incluido (gracias a los nombres largos de Windows 95/98 o Windows NT) y que además se pueden configurar para bajar imágenes, archivos de sonido, etc. La ventaja de los *Offline Browsers* es clara: reducen la factura telefónica al bajarse de una vez los archivos y dar la posibilidad de consultarlos sin estar conectado a Internet.

- **2.Creación de Agentes de Verificación de Enlaces**: aquí los *Robots* cumplen la función de mantener al día los enlaces de un sitio *Web* e informar de los que han cambiado de sitio o los que ya no hacen referencia a ninguna página. Son muy útiles como herramienta administrativa para nuestro propio servidor *Web*.

- **3.Creación de Canales Activos**: al igual que los canales de *Internet Explorer* o de *Netscape Communicator*, un *Robot* puede efectuar la labor de mantener al día un canal, de acuerdo a la frecuencia de actualización con que lo programemos.

- **4.Creación de Buscadores**: Esta es la utilidad que en este momento nos ocupa, la de utilizar el Robot como generador de referencias a páginas *Web*. Hoy día y debido al incremento exponencial de sitios *Web*, los buscadores se han especializado para encontrar datos concretos. Así tenemos entre otros: buscadores de software para buscar la última utilidad de Internet), de noticias (que crean un "periódico" a tu medida), de "lo más barato" (que seleccionan *Webs* y comparan precios de productos semejantes), de Acciones (que actualizan y te envían por correo electrónico las últimas cotizaciones de tus acciones bursátiles y las últimas noti-

cias que afectan a las empresas cotizadas en Bolsa), de personas (que permiten encontrar la dirección e-mail de una persona a partir de su nombre y apellidos), etc.

Un ejemplo de un *Robot Web* se puede ver en la Figura 3. En ella se muestra el buscador basado en el *Robot WebStorm*, que permite realizar búsquedas de igual forma que un buscador basado en *Web*.

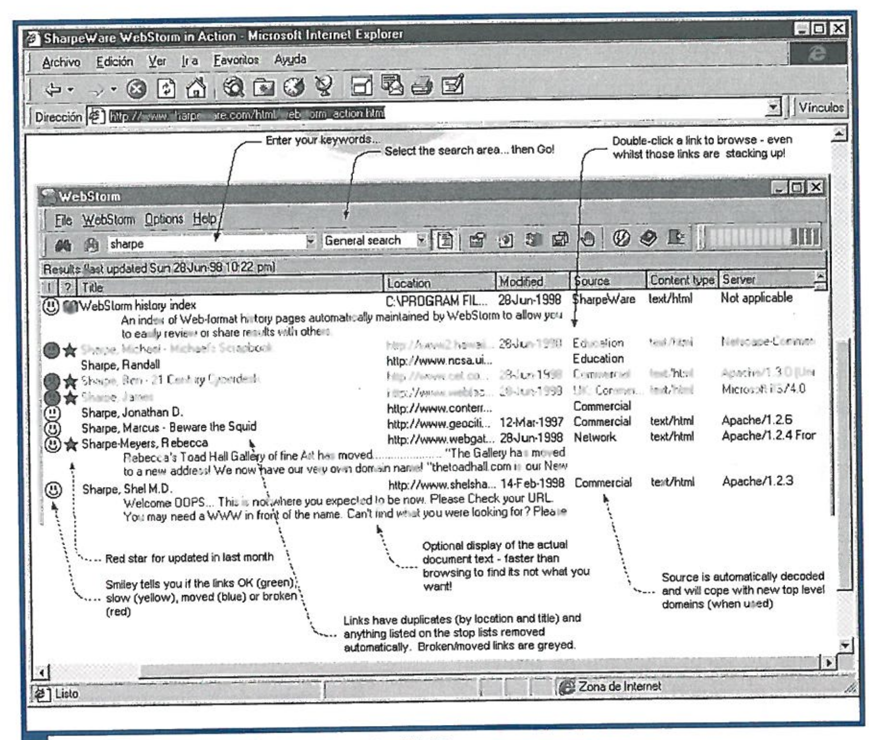


Figura 3. Explicación del Robot buscador WebStorm.

CREACIÓN DE UN BUSCADOR

La creación de un buscador basado en *Web* es un proceso que puede estructurarse en varias tareas, de las cuales, en mayor o menor medida, todas tienen importancia si se quiere conseguir un producto de calidad:

- **1.Creación de un Robot Web**, que a partir de una página analice los enlaces e indexe las páginas encontradas.
- **2.Análisis de los datos que recoge el Robot:** de forma que podamos realizar una preselección "inteligente" de las páginas que pueden resultar interesantes con respecto a las demás, en función de unos criterios determinados.
- **3.Estructuración de los datos en una base de datos,** donde las referencias se almacenen con una lógica prefijada. El diseño de la misma será un punto muy importante también, pues debemos prever los posibles campos de búsqueda y las palabras clave que almacenaremos en cada uno de ellos.
- **4.Diseño de las consultas a la base de datos:** de forma que el resultado de una búsqueda esté de acuerdo con lo que el usuario está solicitando (que se comporte de la forma lo más "inteligente" posible).
- **5.Creación de la aplicación de servidor que recoge la petición del cliente:** es decir, la aplicación encargada de analizar la petición, lanzar la consulta a la base de datos y enviar la respuesta al servidor *Web*, para que éste se la envíe al cliente.
- **6.Creación de los formularios HTML** de búsquedas normal y avanzada, mediante los cuales enviaremos a través del Navegador la petición al servidor *Web*.

EXCLUSIÓN DE ROBOTS

Debido a que los *Robots* funcionan de forma automática, se detectó, desde que nacieron los primeros, un problema de sobrecarga que producían en la Red. La razón es que si estaban mal diseñados, generaban sin parar tráfico de páginas *Web*, sin otro fin que el de indexarlas. Del propio problema se obtuvo la solución: otorgar en los servidores *Web* la posibilidad de no permitir el acceso a determinados *Robots* o no permitir la inspección de determinadas rutas de dicho servidor.

La creación de un Robot Web es sólo uno de los pasos necesarios para crear un buscador Web

Los *Robots Web* proporcionan mecanismos de "privacidad" de forma que los sitios *Webs* que lo deseen no sean inspeccionados (lógicamente al proporcionar este mecanismo a los propios *Robots*, su eficacia dependerá de la buena voluntad de estos últimos).

El método empleado para que un servidor *Web* evite la inspección del

mismo por parte de los *Robots* se estructura en dos ámbitos: por un lado, se facilita al Administrador del *Web* un mecanismo de exclusión de Robots y por otro, se proporciona al propietario de cada página *HTML* un mecanismo adicional de control del acceso a la misma por parte de los *Robots*.

El primer mecanismo se denomina "Protocolo de Exclusión de *Robots*", y permite al Administrador decidir qué partes del *Web* no deben ser indexadas. El medio para conseguirlo: un archivo de texto denominado **ROBOTS.TXT** que contiene las instrucciones sobre las páginas visitables y las que no permiten el acceso a los *Robots*.

El segundo mecanismo, que proporciona un nivel adicional de protección para el propietario de cada página, se logra mediante la inserción de unas etiquetas *HTML* denominadas **META Tags** (es decir: Etiquetas *META*) en las que se indica al *Robot* si debe o no inspeccionar o indexar cada página *HTML* individual.

Es importante recordar que estos métodos de protección frente a los *Robots* dependen del buen comportamiento de los mismos, ya que resulta trivial para un *Robot* saltárselos a la torera.

EL FICHERO ROBOTS.TXT

El *"Protocolo de Exclusión de Robots"* se basa en la especificación del contenido de un fichero **ROBOTS.TXT** en el que se presentan las instrucciones de comportamiento oportunas para los *Robots* en relación con las páginas inspeccionables. La razón de elegir un fichero como método de exclusión de páginas es que con sólo "bajarse" este fichero, el *Robot* conoce las páginas indexables de un servidor.

Dicho fichero debe residir en el directorio raíz del servidor *Web* (es decir, en el directorio donde se encuentra la página por defecto). Por ejemplo, en el *Web* de la revista: **http://www.towercom.es** el archivo **ROBOTS.TXT** sería accesible a través del siguiente URL:

http://www.towercom.es/robots.txt

Este fichero está compuesto por una serie de registros (o puede estar vacío) separados por líneas en blanco (con saltos de línea). Cada registro tiene el siguiente formato:

DIRECTIVA:< espacio opcional > VALOR <espacio opcional>
Comentario

El nombre de la directiva puede ir en mayúsculas o minúsculas indistintamente. Las directivas posibles son *User-agent* (agente de usuario) y *Disallow* (no permitir). Cada registro comenzará con una o más líneas con directivas *User-agent*, y a continuación, una o más líneas con directivas *Disallow*. El significado concreto de cada una de estas dos directivas es el siguiente:

- **User-agent** (agente de usuario): Define, para el registro en el que se encuentre, el nombre del *Robot* para el que se dan normas de acceso. Se pueden definir varios agentes en el mismo registro si las normas son las mismas para todos. Si el valor es "*", significa que el registro describe reglas de acceso para el resto de los *Robots* que no han sido incluidos en otros registros. Por tanto, sólo un registro del fichero **ROBOTS.TXT** podrá indicar como "*" el *User-agent*".

- **Disallow** (no permitir): Define la *URL* que no debe ser inspeccionada por el *Robot* o *Robots* indicados en *User-agent*. La *URL* puede ser parcial o completa. Por ejemplo:

```
Disallow: /doc
```

No permite el acceso ni a /doc.html, /doc/doc1.htm, /doc/doc2.htm,etc.

```
Disallow: /doc/
```

No permite el acceso a todos los archivos del directorio /doc pero sí permite el acceso a /doc.htm".

Si el valor de la directiva está vacío significa que el *Robot* indicado en el registro puede inspeccionar todas las *URL's*.

Ejemplos de ficheros **ROBOTS.TXT**:

- 1.Si no queremos que ningún *Robot* inspeccione nuestro sitio *Web*, el fichero deberá contener lo siguiente:

```
User-agent: *
Disallow: /
```

- 2.Si queremos impedir que cualquier *Robot* visite cualquier *URL* que comience por **/doc/confidencial/** y **/doc/personal/**, deberá contener un registro con las siguientes directivas:

```
User-agent: *
#documentos confidenciales de la empresa
Disallow: /doc/confidencial/
#documentos personales
Disallow: /doc/personal/
```

- 3.Si se quiere conseguir lo mismo que en el ejemplo anterior excepto para un *Robot* concreto (por ejemplo **robot_solop**), el fichero será:

```
#todos los Robots tienen restringido el acceso.
User-agent: *
Disallow: /doc/confidencial/
Disallow: /doc/personal/

#el Robot: "robot_solop" tiene permitido el
    acceso.
User-agent: robot_solop
Disallow:
```

ETIQUETA META PARA ROBOTS

Aunque no todos los *Robots* la soportan la etiqueta *META* proporciona el medio de indicar a los *Robots* visitantes si la página puede ser indexada o si los enlaces de la página pueden inspeccionarse. Se diferencia del "Protocolo para Exclusión de Robots" en que no es necesario obtener el permiso del Administrador del servidor *Web*.

La etiqueta *META* para *Robots* está compuesta de un conjunto de directivas separadas por comas:

```
<META NAME="ROBOTS"
      CONTENT="ALL | INDEX | FOLLOW
      | NONE | NOINDEX | NOFOLLOW">
```

Por defecto, la directiva *CONTENT* está vacía y significa *ALL*.

El campo *ALL* agrupa *INDEX* y *FOLLOW*.

El campo *NONE* agrupa *NOINDEX* y *NOFOLLOW*.

El carácter "|" indica que los campos son opcionales.

Las directivas pueden ir en mayúsculas o minúsculas indistintamente.

La etiqueta *META* se utiliza por aquellos usuarios normales de un servidor *Web* que no tienen acceso al fichero **ROBOTS.TXT** (por ejemplo, cuando colgamos las páginas web en nuestro proveedor de acceso a Internet). Por tanto, concede al dueño de las páginas una oportunidad de impedir a los *Robots Web* que las inspeccionen (independientemente de lo que diga el fichero **ROBOTS.TXT** del servidor).

Las directivas significan lo siguiente:

- **INDEX** (Indexar): significa que se permite a los *Robots* incluir la página *HTML* en el buscador.
- **FOLLOW** (Seguir): se permite a los *Robots* seguir enlaces de esta página para localizar otras páginas.
- **ALL** (Todo): agrupa las dos anteriores y, por tanto, permitimos al *Robot* tanto indexar la página como explorarla.
- **NOINDEX** (No indexar): Aunque no deseamos que se indexe esta página, permitimos que el *Robot* explore sus enlaces.
- **NOFOLLOW** (No seguir): No permitimos explorar los enlaces de la página, pero permitimos que la página sea indexada.
- **NONE** (Nada): significa que el *Robot* debe ignorar esta página.

De esta forma, las posibles combinaciones son las siguientes:

```
<meta name="robots" CONTENT="ALL">
<meta name="robots"
      CONTENT="INDEX,FOLLOW">
<meta name="robots" CONTENT="NOIN-
      DEX,FOLLOW">
<meta name="robots"
      CONTENT="INDEX,NOFOLLOW">
<meta name="robots" CONTENT="NONE">
<meta name="robots" CONTENT="NOIN-
      DEX,NOFOLLOW">
```

La etiqueta *META* se puede utilizar conjuntamente con otra etiqueta META, *DESCRIPTION* (descripción), con la que informamos al *Robot* del resumen textual que queremos que aparezca en el Buscador cuando éste muestre la página en el resultado de una búsqueda.

```
<META NAME="DESCRIPTION" CON-
      TENT="Explicación del contenido de la
      página">
```

¿Y dónde deberían ir situadas estas etiquetas?. El mejor sitio es sin duda la sección *<HEAD>* de la página *HTML*:

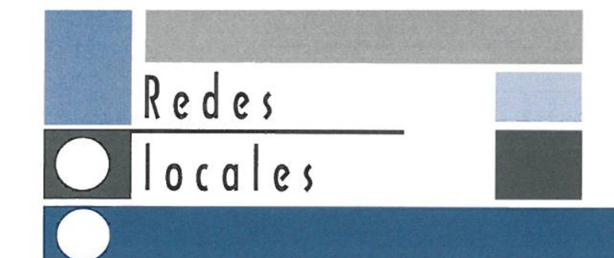
```
<HTML>
<HEAD>
 <META NAME="robots"
      CONTENT="INDEX,FOLLOW">
 <META NAME="description" CON-
      TENT="El proceso de creación de un
      buscador Web se puede estructurar en
      varios pasos: crear un Robot Web, alma-
      cenar en una base de datos la información
      que el Robot recopila, establecer meca-
      nismos de búsqueda en esa base de datos
      a través de un formulario HTML y por
      último, publicarla en un servidor Web.">
  <TITLE>
  Cómo crear un buscador Web
  </TITLE>
 </HEAD>
 <BODY>
 ...
 </BODY>
 </HTML>
```

CONCLUSIÓN

Y aquí terminamos este primer artículo sobre cómo crear un buscador *Web*. Hemos visto las nociones básicas de funcionamiento de un *Robot Web*, las posibles aplicaciones de los mismos y la implementación particular que deberá tener dentro del proceso de creación de un buscador.

Asimismo, hemos estudiado los mecanismos de exclusión de *Robots* que se han desarrollado para evitar en cada servidor *Web*, la indexación de determinadas páginas, y que todo *Robot Web* debería respetar.

Durante los próximos artículos iremos implementando nuestro propio *Robot* para poderlo integrar en el desarrollo conjunto que supone el proceso completo de creación de un Buscador *Web*.



Interconexión de redes (y II)

Enrique Díaz Trobo (enriqued@jet.es)

Continuamos en este artículo viendo aquellos dispositivos que sirven para interconectar redes, haciendo una aproximación somera sobre cómo funcionan internamente y cuál es más adecuado para cada situación, en función de aspectos como el costo de la transmisión, la rapidez, la fiabilidad, etc.

En artículos anteriores hemos visto diferentes elementos para la interconexión de redes, pero dejamos algunos otros en el tintero. Tenemos otros dispositivos que trabajan en niveles superiores del modelo *OSI* y por tanto ofrecen unas prestaciones mayores que los ya vistos.

MAU'S

MAU significa *Multistation Access Unit* (Unidad de acceso multiestación). Se trata básicamente de un *hub* que es capaz de implementar un anillo lógico a través de una topología en estrella física. En la mayoría de los casos,mientras se está hablando de topología en anillo, no quiere decir que se hable de la forma física del circuito que forma el cable de la red, sino que más bien se hace referencia al circuito lógico que sigue un paquete transmitido a través de la red, independientemente de la forma física que tenga este circuito.

También tenemos lo que se llama *anillo colapsado* (traducción horrible donde las haya, por cierto). Pese a lo que su traducción indica, se trata de una individualización del término soporte reducido. En realidad significa soporte reducido en tamaño y adaptado a una única caja.

Imaginemos que tenemos una red con topología en bus y que reducimos el cable al que están conectados todos los ordenadores hasta ser capaces de meterlo en una caja a la que conectamos los ordenadores (aquí ya con cables normales y no miniaturizados). Diríamos entonces que tenemos un bus reducido (o colapsado). Del mismo modo, un anillo colapsado o reducido se puede implementar a través de una *MAU*.

En artículos anteriores vimos uno de los contras de las redes en anillo: si se rompe el cable, y por tanto se corta la comunicación en algún punto del anillo, toda la red quedaba paralizada. Por este motivo se implementa físicamente como una estrella.

Aquí probablemente nos estamos liando ya con *MAUs* y *hubs*. Veamos: Una *MAU* viene a ser un *hub*, e incluso su apariencia física externa es muy similar, pero la cuestión se encuentra en la forma en que está estructurado internamente. Mientras que un *hub* propiamente dicho interconecta de forma interna todos sus puertos para crear un bus lineal, una *MAU* implementa internamente un anillo.

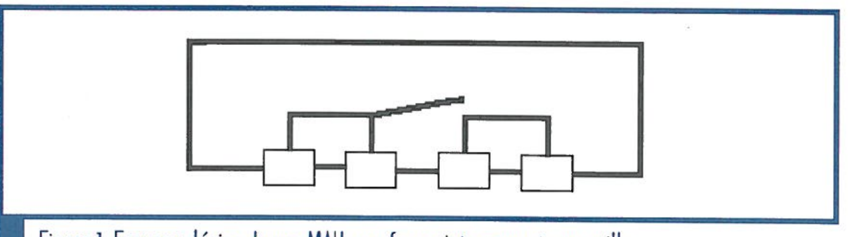
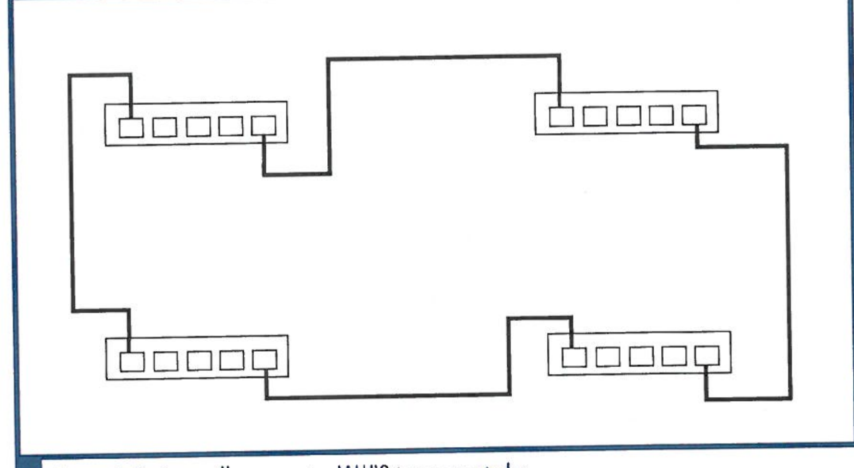


Figura 1. Esquema lógico de una MAU, que forma internamente un anillo.

La clave reside en que una *MAU* puede hacer que cuando se rompe el anillo en algún punto, o bien alguna estación conectada a él falla, automáticamente desconectará de forma lógica la estación que falla manteniendo así el anillo cerrado para que todo funcione de manera correcta.

Para añadir una estación a un anillo basta con conectar la estación a un puerto libre de la *MAU*, de tal forma que ésta realizará una extensión del anillo lógico, consiguiendo así ampliar el anillo con la nueva estación. Cuando se acaben los puertos de la *MAU* podremos añadir otra conectando la entrada de la primera *MAU* a la salida de una segunda y de esta forma cerrar el anillo (figura 2).


Figura 2. Red en anillo con varias MAU'S interconectadas.

ENLACE BACKBONE

Un enlace central (*backbone*) es una red que conecta dos o más segmentos de red local o subredes y se emplea generalmente para distancias medias o largas, entre plantas de un edificio, por ejemplo. Cada red se conecta al *backbone* mediante un *bridge* (puente) o *router* (encaminador) , pero los *routers* resultan más adecuados si la red es grande y necesita conectar muchos tipos diferentes de topologías de red. Si conectamos mediante *bridges* a un *backbone* entonces haremos que todas las redes conectadas compartan la misma dirección, a no ser que se utilice un *bridge* con traducción (que en definitiva sería lo mismo que un *router*).

Para sacar todo el partido a los *backbones* se implementa una topología de red rápida, de modo que se pueda gestionar tráfico entre redes diferentes. Por lo general, se emplea como *backbones Fast Ethernet*, *FDDI* (*Fider Distributed Data Interface*), y conmutadores *ATM* (*Asynchronous Transfer Mode*). Los enlaces centrales pueden gestionar el tráfico entre redes, mientras que el tráfico local se gestiona dentro de las propias subredes.

La figura 3 muestra un enlace central: cada servidor contiene dos placas de red. Una de las placas conecta el servidor al enlace central y la otra conecta la red local al servidor. El servidor funcionará entonces como un *router*, enviando el tráfico a las otras redes a través del enlace central. También podemos ver como el enlace central interconecta los segmentos de red en cada planta de un edificio de oficinas. El cable de enlace central se extiende a través de una instalación recorriendo todas las plantas.

La figura 4 muestra un diseño de red centralizada que puede ofrecer una mejor gestión de los recursos de la red. Los servidores departamentales son trasladados a un área central de gestión y conectados a una topología de red rápida como *Fast Ethernet*. Las redes departamentales se conectan al enlace central con *bridges* o *routers*.

De nuevo tenemos aquí la figura del enlace central reducido, esto es, un enlace central de pequeño tamaño para entrar en una caja. En vez de desplegar el cable del enlace central entre las plantas instalamos un *hub* en un punto central y luego llevamos los cables a los *hubs* de cada planta hasta el *hub* central. Puede parecernos que esto

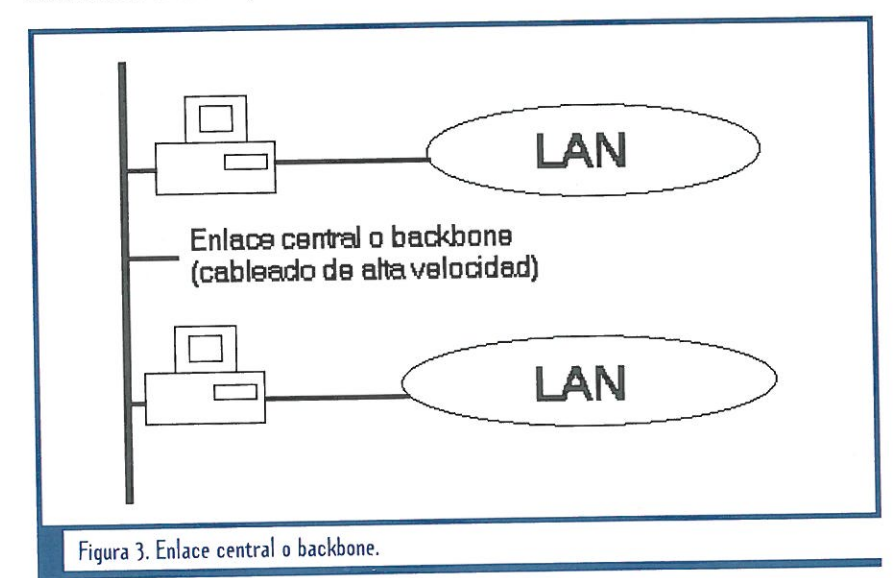


Figura 3. Enlace central o backbone.

requiere más cableado, pero por lo general este esquema ofrece una gran tolerancia a fallos.

Mientras que los *backbones* compactos y sistemas de cableado estructurado ofrecen para redes medias (*LAN* y *MAN*), las *WAN* deberán crear 'enlaces centrales' entre grandes áreas. Los servicios como *Switched-56*, *Frame Relay*, *Switched Multimegabit Data Service* (*SMDS*) y *ATM* (Modo de transferencia asíncrono) son servicios de alta velocidad que ofrecen ancho de banda variable y pago según utilización.

Un enlace en *backbone* se utiliza, entre otras cosas, para estructurar una red de forma que podamos segmentarla en función de algún criterio: por ejemplo, si tenemos una red distribuida en un edificio se puede crear un segmento de red por planta e interconectarlos a través de un cable de alta velocidad con enlace *backbone* que pase por todas las plantas del edificio.

Otra de las aplicaciones para las que se utiliza un enlace *backbone* es para centralizar los servidores de una red, donde cada uno de ellos gestiona su segment. Con esta configuración se interconectan los servidores a través de un enlace *backbone*. Las ventajas de centralizar los servidores son muchas, entre otras facilitar su mantenimiento, mejorar su seguridad física, centralizar operaciones propias de los servidores; copias de seguridad, comprobación de la red, etc.

HUBS

Los *hubs* o centralitas de cableado son centros de control que se utilizan para gestionar y comprobar el cableado de una red distribuida por un edificio. Se trata de un dispositivo con múltiples bases para módulos capaz de albergar dispositivos de diferentes tipos, como concentradores, *bridges*, etc. Es decir, una centralita de cableado es una plataforma para la interconexión de diferentes tipos de módulos de comunicaciones para redes. Estos concentradores están compuestos por una fuente de alimentación (a veces dos, por si una de ellas falla) y un bus de expansión al que se conectan diferentes dispositivos.

EVOLUCIÓN DE LOS HUBS

Las redes locales Ethernet *originales* se crearon llevando cables entre edificios y conectando cada estación punto a punto. Cuando todas las estaciones estaban conectadas, se colocaba un terminador en cada extremo del cable y se arrancaba la red. Había muchas cosas que podían evitar que toda la red funcionara a la primera, tal como conectores mal fijados (quien se haya peleado alguna vez con los conectores *BNC* lo comprenderá a la perfección), interferencias de fuentes externas, bucles de tierra y cables doblados o pisados, rebotes de la señal.

Una MAU implementa interiormente un anillo, mientras que un hub implementa una red en bus

A falta de un buen programa de comprobación de las comunicaciones, lo que se solía hacer (y aún se hace) es colocar el terminador de red de forma que resulte una red de dos estaciones, arrancar la red y ver si funciona. Si funciona, ampliar la red a tres terminales; y así hasta que se descubría la estación que fallaba.

Las topologías basadas en *hubs* fueron diseñadas precisamente para ayudar a resolver estos problemas. Los *hubs* han evolucionado en varias generaciones, siendo los primeros simples *hubs* repetidores. Ahora son los componentes principales en los sistemas de cableado estructurados que soportan muchas topologías de redes locales y de gran alcance. Un *hub* puede servir como centro de conexiones para toda una planta, un edificio, un campus, etc.

Los *hubs* se hicieron muy comunes en el entorno de redes locales con el desarrollo de topologías *10Base-T* configuradas en estrella.

La siguiente generación consistió en desarrollar una caja con conectores de expansión a los que se pudieran añadir módulos multipuerto a medida que fuera necesario. Este diseño de *hub* multiconector en una sola caja centraliza las fuentes de alimentación y componentes, y reduce su coste.

Los *hubs* inteligentes han supuesto una pequeña revolución y destacan por incluir funciones de administración, sistemas de detección de fallos y módulos para enlace de encaminamiento. También proporcionan funciones para recoger estadísticas sobre los módulos de los *hubs* en cada uno de sus puertos, tratados de forma individual. Podemos gestionar los *hubs* desde una estación remota utilizando protocolos de gestión *SNMP*.

Una característica muy interesante consiste en que podemos crear segmentos lógicos de una red local dentro de un único *hub*, esto es, crear redes virtuales. Esta característica permite dividir una red local en segmentos más pequeños por razones de organización y rendimiento. También constituye una gran ayuda en cuanto a seguridad, si pretendemos exponer parte de nuestra red a Internet u otras redes. De este modo, y sin complicarnos mucho la vida podemos exponer sólo una parte de una única red física al exterior.

Otra generación más la constituyen los *hubs* llamados de tercera generación, los cuales se caracterizan por su orientación hacia grandes empresas, diseñados para soportar todo el cableado y necesidades de conexión entre redes de una gran organización.

Incorporan características inteligentes, *backplanes* de alta velocidad y gran modularidad, soportando un gran número de módulos, incluyendo conexiones de largo alcance y gestión avanzada.

Estos sistemas tienen diseños de buses de gran velocidad para gestionar todo el tráfico de la empresa, y características avanzadas de gestión para realizar una comprobación continua de la red e informar de su estado. La fiabilidad también tienen gran importancia y existen muchas características redundantes para protegerse contra sí mismos. Es decir, protegerse contra fallos de los componentes, como la fuente de alimentación, el bus y enlaces de gran alcance.

Muchos de estos nuevos *hubs* utilizan *backplanes* de conmutación de celdas *ATM Asynchronous Transfer Mode* (aquí ya estamos hablando en *gigabits*). En resumen estos concentradores aportan:

- *Backplanes* segmentados para soportar varias redes, como *Ethernet*, *Token Ring* y *FDDI*.
- Encaminamiento y enlace a alta velocidad entre redes.
- Capacidades de conmutación para microsegmentar la red.
- Circuitos dedicados entre nodos terminales para soportar tráfico de alto volumen o de respuesta rápida.
- Prestaciones de gestión distribuida incluidas en cada módulo, para mejorar el rendimiento bajo condiciones de carga muy alta.

TIPOS DE HUBS

Los *hubs* se clasifican en función de cómo se utilizan en un sistema de cableado estructurado. Atendiendo a esta clasificación tendremos tres categorías: el *hub* de grupo de trabajo, el intermedio y el de empresa.

- *El hub de grupo de trabajo* consiste en un *hub* que conecta un grupo de equipos que se encuentran físicamente cercanos; es la típica red que está en una sola habitación.
- *Los hubs intermedios* son los que se encuentran generalmente en la centralita de conexiones de cada planta. Es allí donde se conectan los *hubs* de grupo de trabajo. Estos *hubs* pueden estar a su vez conectados al de la empresa. El tráfico entre los *hubs* de grupos de trabajo locales puede gestionarse a través del *hub* intermedio, o bien el *hub* de empresa puede gestionar todo el tráfico de la interconexión de redes, dependiendo de las necesidades o bien del propio diseño de los *hubs*.
- *Los hubs de empresa* constituyen el punto de conexión central para todos los sistemas finales conectados a grupos de trabajo. Forman el propio *backbone* o bien ofrecen conexiones a un *backbone*. Pueden ofrecer *bridges*, encaminamiento y servicios de

conexión de gran alcance. El *hub* de empresa puede incorporar módulos de gestión avanzada.

Por supuesto, todo lo descrito es de carácter modular, ya que podemos comenzar a constituir una red con los *hubs* de grupo de trabajo, y posteriormente añadir otros intermedios o de empresa a medida que vayamos creciendo. Los *hubs* de empresa tienen que estar diseñados para gestionar los requisitos típicos de organismos concretos y deben ser compatibles con las nuevas tecnologías *ATM*.

CARACTERÍSTICAS DE GESTIÓN

Además de las mejoras en el ámbito físico, un aspecto muy importante lo constituyen las características de gestión. La mayoría de los *hubs* de alto nivel tienen su propio microprocesador, que puede ejecutar programas para controlar paquetes de datos y errores, y almacenar esta información en una base de información de gestión llamada *MIB* (*Management Information Base*, no *Men In Black*, como habíais pensado).

Un programa de gestión que se ejecuta en la estación del administrador de la red recoge periódicamente la información de las *MIB* y la presenta al administrador. Esta información resulta útil para controlar tendencias, diagnosticar problemas, mostrar problemas de congestión, etc.

Existen alarmas que pueden avisar a los administradores cuando se alcanzan diversos umbrales que podrían traer problemas a la red. Por ejemplo, se puede producir una alerta cuando el tráfico de la red supera un determinado nivel, de modo que podremos tomar a tiempo medidas correctoras, como segmentar la red local o pasar un usuario que genera un alto volumen de información a un segmento dedicado.

La mayoría de fabricantes soportan el protocolo básico de gestión de red (*SNMP*), que funciona sobre el protocolo *TCP/IP* (*Transmission Control Protocol/Internet Protocol*) y lo utiliza como medio de transporte para obtener así información de la *MIB* y llevarla al ordenador de gestión. Si se alcanzan ciertos umbrales, *SNMP* utiliza *TCP/IP* para enviar los mensajes de aviso al ordenador que lleva la gestión de la red. El *SNMP* también puede desconectar automáticamente los nodos problemáticos que perturben la red, desconectar estaciones en fecha y hora, etc.

REDES VIRTUALES

Existen *hubs* que permiten simular segmentos de red. Las redes locales virtuales facilitan la creación de los grupos de trabajo y pueden mantener el tráfico de un grupo de trabajo dentro de los límites lógicos del mismo. De igual forma constituyen una ayuda para la seguridad y reducen la congestión en la interconexión de redes.

Pero para alcanzar estos objetivos necesitamos controlar el tráfico entre las redes virtuales con técnicas de encaminamiento. El encaminamiento dentro de una red virtual se puede realizar de varias maneras. En el método con enlace central, se colocan *routers* en el enlace central y todo el tráfico de la red circula a través del mismo, incluso si el destino está muy cerca del origen. El encaminamiento se gestiona de forma centralizada.

Los hubs de última generación comprueban la contención en la red e informan del estado en que se encuentra la propia red

En otros métodos, los conmutadores cercanos a las estaciones toman algunas decisiones de encaminamiento, pero no realizan tareas tradicionales de los *routers*, como el cálculo de tablas de encaminamiento. Un servidor de rutas construye todas las tablas de encaminamiento y las envía a los conmutadores. Sin embargo, los conmutadores deben comprender las tablas creadas por el servidor de rutas, por lo que esta técnica puede necesitar que todos los equipos sean adquiridos a un único fabricante, ya que no existe un estándar definido para realizar estas tareas.

HUBS DE CONMUTACIÓN

Un *hub* de conmutación es un concepto relativamente nuevo que saca partido de las topologías en estrella para reducir la contención en segmentos de red *Ethernet*. La conmutación implica la **microsegmentación** de redes locales para que haya un número reducido de estaciones en una red local y por lo tanto se reduzca la contención. Los *hubs* de conmutación originales fueron diseñados para grupos de trabajo, pero hay ya dispositivos de conmutación que son unidades modulares que se conectan a los *hubs*.

Supongamos que tenemos una red que está colapsada por un exceso de tráfico; podríamos optar por segmentar la red y conectarla con un *bridge*. Esto reduciría el tráfico y la contención en cada segmento. Sin embargo no nos garantiza que desaparecerán los problemas de tráfico. Podríamos continuar segmentando la red hasta que desaparecieran los problemas, lo cual podría salirnos bastante caro. Esto es justamente lo que hace un *hub* de conmutación, realizar las segmentaciones sucesivas que sean necesarias hasta que desaparecen los problemas de tráfico. Este tipo de *hub* tiene varios puertos, cada uno de los cuales es un segmento de red diseñado para la conexión de *hubs* de grupos de trabajo o incluso de una única estación. El tráfico lo gestiona un bus interno multiplexado de alta velocidad. La conmutación se gestiona en el subnivel de control de acceso al medio (*MAC*), lo que implica que los

conmutadores son básicamente *bridges* multipuerto.

Lo realmente interesante es que al tener menos estaciones en cada segmento se genera menos tráfico y menos contención. Supongamos que una estación que está conectada a un puerto necesita conectarse a una estación de otro puerto. El *hub* de conmutación conecta los dos puertos, creando de este modo una red de dos nodos sin contención. Así pues, se aprovecha todo el ancho de banda de *Ethernet* para transmitir. El conmutador gestiona el tráfico entre un puerto y otro a través de un bus de datos de alta velocidad, y este bus puede gestionar transferencias simultáneas de datos entre puertos. Para ello, el conmutador construye una tabla con la dirección de control de acceso al medio (*MAC*) de la estación conectada a cada puerto.

Una red *Ethernet* conmutada se actualiza fácilmente desde las instalaciones *10Base-T* y no requiere cambiar las tarjetas de red de las estaciones ni modificar el cable existente de par trenzado. De este modo, con cambiar los *hubs* por otros que ofrezcan conmutación se soluciona el problema.

BRIDGES

Son dispositivos inteligentes capaces de conectar dos segmentos de red, incluso si estos no son iguales, es decir, si tienen diferentes métodos de control de acceso al medio, por ejemplo podemos conectar un segmento en bus y otro con *Token Ring*.

Los puentes trabajan a nivel de enlace. Cualquier dispositivo que cumpla las especificaciones del subnivel de enlace *MAC* (*Media Access Control*) puede conectarse con otros dispositivos que igualmente las cumplan gracias a un *bridge*. Al trabajar a nivel de enlace, solamente entienden de direcciones *MAC*, y no otro tipo de direcciones como pueden ser las direcciones IP, localizadas en el nivel de red, que es el que se encuentra por encima del nivel de enlace.

Un *bridge* viene a ser un clasificador que lee los paquetes de datos que llegan a él, estudia su dirección de red y los envía hacia la red o subred adecuada. No realizan tareas de encaminamiento, es decir, conocen las direcciones de las redes que interconectan y analizan la dirección de red de un paquete que les llega, enviándolo hacia la red destino por el camino que tiene establecido. Sin embargo, no es capaz de estudiar el tráfico de la red en un momento determinado y elegir la ruta que debe seguir el paquete para llegar a su destino, atendiendo a factores como el tiempo invertido en la comunicación, seguridad, fiabilidad, coste, etc. De hecho un puente añade un tiempo de retraso, ya que el paquete transmitido tiene que ser analizado para ver su dirección de red y redirigirlo a su destino.

Los puentes ofrecen funciones de filtrado, que posibilitan que el rendimiento de una red mejore cuando se segmenta a través de un puente. La función de filtrado consiste en no dejar pasar a segmentos de red paquetes con información que no van destinados a ese segmento.

A un puente le pueden llegar paquetes de diferentes naturalezas: de información de solicitud de ruta, de respuesta, etc. Tanto el puente como las estaciones distinguen estos paquetes mediante unos campos de control introducidos en los paquetes transmitidos.

Existen muchos tipos de puentes: puentes de aprendizaje, puentes *tandem*, con distribución de carga, transparentes, adaptativos, de carga equilibrada, locales, remotos etc. Las clasificaciones de los puentes se hacen atendiendo a diferentes aspectos. Además, los puentes pueden materializarse como dispositivos externos, o bien a través de software, instalando varias tarjetas de red en un servidor, siempre y cuando dicho servidor contemple la posibilidad de actuar como puente en el software que incorpora.

Una primera clasificación se podría efectuar en función del tipo de elementos que interconectan; los puentes locales son los que interconectan segmentos de una red local, o bien redes locales distribuidas en un edificio, mientras que los puentes remotos interconectan redes lejanas ente sí (redes *WAN*).

Una segunda clasificación hace referencia al algoritmo seguido para localizar el destino de un paquete: puentes transparentes o adaptativos y puentes de encaminamiento fuente.

PUENTES TRANSPARENTES O ADAPTATIVOS

Los puentes transparentes son más complejos que los de encaminamiento fuente, pues deben ser capaces de deducir a partir de unas tablas de encaminamiento la dirección de la máquina destinataria del paquete recibido. Las tablas se van actualizando dinámicamente, añadiéndose en ellas nuevas direcciones.

El puente asocia a cada puerta de entrada con una tabla en la que se apuntan las direcciones de todos los paquetes que entran por dicha puerta. De esta forma genera una tabla por cada segmento de red. Cuando le llega un paquete por una puerta analiza si tiene apuntada su dirección; si no la tiene actualiza la tabla correspondiente con dicha puerta y a continuación analiza la dirección destino del paquete y la busca en las tablas. En caso de encontrarla envía el paquete por la puerta correspondiente a la tabla en la que la ha buscado, a excepción de si coincide con la puerta por la que llegó el paquete. En este caso supone que la trama llegó a su destino antes de llegar al propio puente, por lo que la descarta la trama.

Si no la encuentra, inicia un proceso para descubrir el destinatario del paquete, reteniendo el paquete de información que le ha llegado hasta que

conozca el segmento de red al que debe enviarlo. Para ello, manda un paquete de control (diferente del paquete de información que le ha llegado) a todos los segmentos de red, a excepción del segmento al que pertenece la máquina origen. Cuando este paquete de control llega a la máquina destino, ésta responde con otro paquete de control que es analizado por el puente, deduciendo de esta forma el segmento de red al que pertenece la máquina destino. De esta manera envía finalmente el paquete de información. Además, apunta esta nueva correspondencia para próximas ocasiones.

Estas tablas pueden llegar a ser realmente voluminosas si las redes que se conectan son grandes e incluso si se conectan de esta manera muchas redes pequeñas. Tener unas tablas grandes ocasionaría que la búsqueda en ellas supusiera un tiempo de retardo considerable en la transmisión del paquete. Para subsanar esta situación existen diferentes métodos de optimización de las tablas, como puede ser el eliminar correspondencias que hace mucho tiempo que no son utilizadas para reducir el tamaño de las tablas. Además, los algoritmos de búsqueda en dichas tablas suelen estar implementados a través de hardware, que siempre es más rápido.

Los puentes transparentes se suelen utilizar para segmentar una red de tipo 802.3, y para interconectar redes 802.3 y redes en anillo.

PUENTES DE ENCAMINAMIENTO FUENTE

Los otros tipos de puentes son los que emplean el algoritmo de encaminamiento fuente; estos puentes son más sencillos, ya que ellos sólo deben leer la ruta descrita en el paquete para enviarlo por la puerta indicada. Con este tipo, son las propias máquinas las que deben introducir en el paquete no sólo la dirección destino sino además la ruta que debe seguir para alcanzar su destino. Por tanto, son las propias máquinas las que deben contener las tablas de encaminamiento.

El modo de funcionamiento en este caso es el siguiente: cuando una estación quiere enviar un paquete a otra, busca la ruta para llegar a la estación destino en su tabla; si la encuentra copia esta ruta en el paquete de información a transmitir y lo envía. Si no la encuentra, entonces la estación origen envía un paquete de difusión para localizar a la estación destino, el cual durante su viaje va apuntando todos los puntos por los que pasa, es decir, va construyendo la ruta que se deberá seguir para llegar al destino.

Cuando dicho paquete llega a un puente se multiplica saliendo de esta forma por todos los segmentos o redes que interconecta. Es posible que el paquete llegue varias veces a su destino ya que cuando se interconectan redes pueden existir varios puntos de entrada para llegar a una estación, y por tanto, varios caminos por los que un paquete puede llegar de una red a otra. Si existen varías rutas para alcanzar el destino, será la propia estación de destino la que analice todas las rutas propuestas y se selecciona la más idónea, basándose en una serie de factores, y descartando el resto.

Una vez elegida la ruta, el puente reenvía el paquete de control con esta ruta a la estación que lo originó. La estación que envió el paquete extraerá la ruta de este paquete de control y la copiará en el paquete de información que quería enviar. Por supuesto, también apuntará la ruta para emplearla en futuras ocasiones.

Las ventajas que este método ofrece se centran fundamentalmente en la liberación de trabajo al puente, que únicamente tiene que leer la ruta y redirigir el paquete hacia el destino por el camino indicado. Por lo tanto, los tiempos de retardo que introduce son menores.

Este tipo de puente se emplea mucho en las redes *Token Ring*, generalmente sólo con propósitos de segmentación. Los puentes tanto de encaminamiento fuente, como los transparentes presentan una serie de inconvenientes:

- No diferencian entre clases de servicio (por el nivel al que trabajan), porque no son capaces de dar prioridad a unos paquetes sobre otros.
- No existe control de tráfico, por lo que un puente debe reenviar todos los paquetes que recibe o eliminarlos, ya que no puede pedir a las estaciones que bajen el ritmo de transmisión de paquetes hacía él, con lo que el puente podría llegar a saturarse.
- El número de paquetes de control (de exploración y respuesta) aumenta el tráfico de la red, por lo que disminuye su rendimiento.

Además, como ya hemos dicho, los puentes pueden implementarse dentro de un servidor que sea capaz de gestionarlos, como pueden ser estaciones UNIX o servidores *Novell*. En estos casos, para implementar el puente necesitamos instalar en el servidor tantas tarjetas de red como segmentos o redes se necesite conectar. Evidentemente dichas tarjetas deben corresponder con la de los tipos de redes o segmentos que se interconectan. Además, la máquina debe tener el software necesario para gestionar este puente.

En el caso de *Novell* más que un puente tendremos prácticamente todo un *router*, esto es, añadimos a las características propias de un puente funciones de filtrado y la capacidad de trabajar con diferentes protocolos (*IPX/SPX*, *TCP/IP*) sobre uno o varios segmentos de la red.

INTERCONEXIÓN DE PUENTES

Existe un problema importante cuando se quieren interconectar puentes

transparentes con puentes de encaminamiento fuente, ya que las estaciones conectadas al puente transparente no entienden de tablas de encaminamiento. Si queremos interconectar diferentes segmentos de red a través de puentes, podemos establecer diferentes topologías dependiendo de características tales como la necesidad de evitar problemas en las conexiones, interconexión de varios segmentos distantes, etc. Atendiendo a estas características podemos clasificar las topologías con puentes en: topología simple, paralela, en serie y en bucle.

- *Simple*: aquella que conecta dos segmentos de red a través de un único *bridge*.
- *Paralelo*: se basa en la conexión de dos segmentos de red con dos *bridges*. Los dos ofrecen las mismas conexiones, pero si uno de ellos falla el otro sigue haciendo posible la comunicación entre los dos segmentos. Este tipo de topología se utiliza cuando se quiere tener mayor seguridad en las comunicaciones.
- *Serie*: conecta varios segmentos de red, intercalando entre cada dos segmentos un *bridge*.
- *Bucle:* se caracteriza por tener puentes conectados a segmentos formando entre todos un bucle cerrado. El número de segmentos mínimos en esta configuración es de tres. Se suelen utilizar fundamentalmente con topologías en anillo.

ROUTER

Los *routers* o encaminadores son dispositivos que trabajan a un nivel superior a los puentes, es decir, trabajan en el nivel de red y por tanto en el nivel en el que se encuentra el protocolo *IP*. Ofrecen, por tanto, mayores prestaciones que los puentes, pero también son bastante más complejos.

Un *router* interconecta varias redes, de forma que cuando un paquete tiene varios caminos para llegar a su destino, éste es capaz de seleccionar el mejor camino atendiendo a una serie de características (encaminamiento inteligente). El *router* selecciona el camino basándose en unas tablas de encaminamiento, lo que supone una diferencia destacable con respecto a los puentes. Además de esta diferencia, podemos citar otras tan importantes como: capacidad de filtrado avanzado (aislar paquetes entre subredes), capacidad de manejo de varios protocolos (caso de *routers* multiprotocolo), etc.

Cuando un paquete llega a un *router*, éste examina su dirección destino y comprueba sus tablas de encaminamiento. Si puede enviar directamente el paquete a su destino lo hace y en otro caso determina la posición de otro *router* que puede enviar el paquete a su destino. La elección del camino óptimo puede hacerla el responsable de la red o dejársela al *router*. Los factores que se evalúan a la hora de seleccionar el camino óptimo son muchos, entre los que se pueden destacar: camino menos saturado (con menor tráfico), camino más rápido (por el medio de transmisión que lo implementa), camino más corto (por el número de saltos que el paquete debe realizar para llegar a su destino), camino más barato (cuando se utilizan líneas telefónicas o alquiladas).

Los *routers* pueden implantarse para el manejo de un sólo protocolo o de varios (*routers* multiprotocolo). Un *router* para un único protocolo, por ejemplo *IPX*, sólo puede manejar paquetes de ese protocolo, mientras que un *router* multiprotocolo puede gestionar paquetes de diferentes protocolos de forma simultánea. Además permite dividir una red en redes lógicas (segmentos de red), donde cada segmento tiene una dirección diferente al nivel de red (que en el caso de estar trabajando con *IP* a este nivel, sería una dirección de red *IP* diferente). Estos protocolos definen el método por el cual los *routers* pueden comunicarse entre sí y se ejecutan en el *router* construyendo tablas de encaminamiento basadas en el intercambio de información sobre caminos posibles. Existen diferentes protocolos, como son protocolos de vectores de distancia y protocolos de estado de enlaces.

Al igual que en el caso de los puentes, se puede crear un *router* en una red por el sencillo método de poner varias tarjetas de red en un servidor (como en el caso de *Novell*). Pondremos tantas tarjetas de red como segmentos queramos interconectar. Incluso no es necesario crear el *router* en el propio servidor si instalamos el software de *router* en una estación de trabajo y teniendo en cuenta que éstos deben procesar los paquetes con la inversión de tiempo que conlleva, es preferible liberar al servidor de estas tareas.

GATEWAYS

Un *gateway* o pasarela trabaja a nivel aplicación del modelo *OSI*. Propiamente dicho se trata de un dispositivo que puede ser un ordenador que interconecta sistemas que no utilizan los mismos protocolos de comunicaciones, estructura de datos, lenguajes y arquitecturas. Es decir, además de la función de encaminamiento que ofrecen los *routers* incorporan una función de traducción entre protocolos. Por tanto un *gateway* modifica el empaquetamiento de la información recibida y su sintaxis para adecuarse al sistema destino.

En el mundo Internet, actualmente se denomina *gateway* a aquel dispositivo que realiza funciones de traducción entre aplicaciones y no de encaminamiento, dejando esto último a los *routers*.

Programación Distribuida Orientada a Objetos con CORBA (y II)

Antonio Ruiz Cortés (aruiz@lsi.us.es) y José Luis Alvarez Macías (alvarez@uhu.es)

Uno de los aspectos más relevantes, que en estos momentos marca la diferencia entre la norma CORBA y DCOM es la interoperabilidad que ofrece el primero entre objetos distribuidos sobre plataformas distintas, frente a la pobre y endogámica interoperabilidad de DCOM.

INTRODUCCIÓN

En el número anterior se abordó el estado actual de la Programación Distribuida Orientada a Objetos en general, y de la tecnología *CORBA* en particular. En esta entrega, se presenta al lector un ejemplo, muy básico, que muestra como desarrollar aplicaciones distribuidas **multiplataforma** haciendo uso de la norma *CORBA*.

El escenario propuesto para el ejemplo, estará compuesto por una red de ordenadores con sistema operativo Linux en la que existirá un programa servidor implementado en C++, y un programa cliente[1] implementado en Java. El *ORB* utilizado será **ORBacus** de *Object Oriented Concepts*.

El escenario podría haber sido diferente, por ejemplo, *Windows NT* sobre *Alpha Intel*, *Visibroker* como *ORB* y *ADA* y *COBOL* como lenguajes de programación, en cuyo caso, no cambia significativamente el proceso de desarrollo.

Por otra parte, se darán algunas nociones para orientarnos en la búsqueda del *ORB* adecuado a nuestras necesidades. También se indicarán los pasos que seguir para la instalación de este *ORB* y del compilador **gcc 2.8.x** sobre un sistema Linux.

ELECCIÓN DE UN ORB

La norma *CORBA* engloba a un conjunto de especificaciones que describen los aspectos que pretende cubrir todo sistema que se considere compatible *CORBA*. El alcance de éstos se puede resumir en los siguientes puntos:

- Un Lenguaje de Definición de Interfaces: *IDL* (*Interface Definition Language*).
- Una infraestructura de distribución de objetos: *ORB* (*Object Request Broker*).

Tabla 1. ORB's.

Nombre	Lenguajes	Página web
Chorus	C++	www.sun.com
Java IDL1.1	Java	splash.javasoft.com
ORBacus	C++, Java	www.orbacus.com
OrbixWeb 3.0	C++, Java	
VisiBroker	C++ , Java	www.visigeni.com

- Un conjunto de servicios comunes para los objetos: *CorbaServices*.
- Un conjunto de servicios comunes para las aplicaciones: *CorbaFacilities*.
- Un conjunto de servicios especializados en determinados campos: *Domain Services*.
- Una especificación de compatibilidad entre *ORB's*: *IIOP* (*Internet Inter ORB Protocol*).

Abusando del lenguaje, a partir de ahora se denominará *ORB* a todos aquellos productos software que cubren en mayor o menor medida los aspectos anteriores. En este sentido hay que destacar que la funcionalidad y calidad del *ORB* dependerá del fabricante.

Es necesario que el compilador de C++ soporte el manejo de excepciones

Es importante resaltar que a los *ORB's* se les califica de compatibles *CORBA*, pero en ningún caso, hasta ahora, se indica que cumplan la norma en su totalidad. Es decir, no existe ninguna característica del producto que sea contraria a lo establecido por la norma, pero sin embargo muchos aspectos no son tratados. Por ejemplo, hoy por hoy, no existe ningún *ORB* que implemente todos los servicios básicos.

Resulta evidente pues, que una de las decisiones más importantes en el desarrollo de aplicaciones compatibles *CORBA* reside en la elección del *ORB* (*Object Request Broker*), pues el desarrollo de aplicaciones quedará condicionado en función de sus características.

En la actualidad existe un gran número de empresas y organizaciones fabricantes de *ORBs* compatibles *CORBA*, mostrándose en la tabla 1 algunas de los más conocidas.

Son muchos los criterios que tener en cuenta para una correcta elección del *ORB*. Los requisitos del sistema que se va a desarrollar: lenguajes de implementación, plataformas, etc. y el presupuesto con el que se cuenta, son sólo algunos condicionantes que hacen de esta labor una difícil tarea, sobre todo si se tiene en cuenta la amplia oferta de *ORB's* actual y la juventud de la tecnología utilizada.

Las características que se indican en la tabla 1 son muy generales y en la mayor parte de los casos serán suficientes para elegir el *ORB* adecuado. En los casos de desarrolladores con experiencia será necesario tener en cuenta aspectos técnicos de la arquitectura, funcionalidad e implementación de cada uno de los *ORB's*, para disponer de un mayor conocimiento de cara a la elección.

En nuestro caso, se ha elegido *ORBacus* por ser gratuito cuando no se usa con fines comerciales, por estar disponible para C++ y Java, por soportar tres servicios y por el soporte de sus técnicos, que hasta ahora resulta de una calidad y sobre todo rapidez de respuesta inusuales.

ORBacus está desarrollado por la compañía *Object-Oriented Concepts Inc* (**http://www.ooc.com**), y es compatible con la versión 2.1 de la norma *CORBA*. Entre sus características más importantes cabe destacar:

- Soporte para *IDL*.
- Posibilidad de "mapeo" para los lenguajes C++ y Java.
- Incluye los servicios básicos: *Naming*, *Event* y *Properties*.
- Genera documentación en formatos *HTML* y *RTF*.
- Está disponible para diferentes plataformas y sistemas operativos.

La última versión disponible es la 3.1 beta 1, pero está anunciado que en breve (quizás esté disponible cuando lea el artículo) saldrá la definitiva 3.1. En la página web de *OOC* encontraremos los ficheros **OB-3.1b1.tar.gz** y **JOB-3.1b1.tar.gz** que contienen los ficheros fuentes del *ORB* para C++ y Java, así como documentación del producto.

INSTALACIÓN DE ORBACUS EN LINUX

Como ya se ha indicado, el fabricante de *ORBacus* distribuye el código fuente de su *ORB*, de manera que es necesario un proceso de compilación por parte del usuario para obtener un entorno operativo que permita el desarrollo de aplicaciones.

Por otra parte, la instalación de *ORBacus* en la máquina del cliente de la aplicación, que no del desarrollador, tan sólo supone la instalación del *runtime* del *ORB*, que en este caso se reduce a la copia de los ejecutables correspondientes a los servicios utilizados

por la aplicación (nombre, eventos, etc.) y de la librería de enlace dinámico correspondiente al *ORB*, en el caso que se haya optado por este modelo en el proceso de compilación del *ORB*.

A continuación se muestran los pasos para instalar el entorno de desarrollo de *ORBacus* para C++ y Java bajo Linux. Si estas interesado en instalar *ORBacus* en otro sistema operativo (p.e. Windows NT), el proceso resulta bastante similar. Además la distribución incluye documentación sobre su instalación bajo otros sistemas y plataformas UNIX y para Windows NT 4.0.

Para instalar *ORBacus* es necesario:

- 250 Mb de espacio libre en disco.
- Disponer de los fuentes adecuados.
- Compilador de C++ o Java según el caso.

En la versión 3.1 beta1, los fuentes son distribuidos en los ficheros **JOB-3.1b1.tar.gz** y **OB-3.1b1.tar.gz** correspondientes a la versión para Java y C++ respectivamente.

Respecto al compilador de C++, es necesario que soporte el manejo de excepciones. En el caso de Linux y del compilador **gcc** se necesitará como mínimo la versión 2.8.x. En el siguiente punto se indica el proceso de instalación de dicha versión.

INSTALACIÓN DE ORBACUS PARA C++

Los pasos necesarios para su instalación son:

- Descomprimir los fuentes con **tar xvfz OB-3.1b1.tar.gz** y pasar al directorio generado.
- Ejecutar el *script* **runconfig** que detectará las características de nuestro sistema y devolverá un listado con las órdenes que se deberán utilizar para configurar *ORBacus*. En el cuadro 1 se muestran las preguntas que realizará este proceso.

Cuadro 1. Salida del *script* runconfig.

```
* OOC C++ Configurator *

Enter 'c' if you use a C shell, or 'b' for a bourne shell: [b]

Please select from the following C++ / operating system combinations:

(1) SGI C++ 7.1                          SGI Irix 6.2 or 6.3
(2) SGI C++ 7.2.x                        SGI Irix 6.2, 6.3 or 6.5
(3) SUN C++ 4.1 and 4.2                  SUN Solaris 2.5
(4) HP aC++ A.01.12                      HP-UX B.10.20
(5) AIX C Set ++ xlC 3.1.4.6             AIX Version 4.2.1
(6) DEC C++ 6.0-010                      DEC OSF/1 V4.0
(7) EGCS C++ 1.0.x / GNU C++ 2.8.x       Any supported OS

Please choose your platform/compiler combination: [7]

Do you want to create shared libraries? [no]
Do you want optimized code to be generated? [no]
Add debug information to the generated code? [no]
Compile ORBacus with JThreads/C++ support? [no]
Please enter any extra preprocessor flags, like `-I/some/directory': []
Please enter any extra linker flags, like `-L/some/directory': []
Where do you want to install everything? [/usr/local]

To run `configure', execute the following code in your shell, for
example by using "cut & paste":

  . config.env
  rm -f config.cache
  ./configure
```

- Para poder utilizar *ORBacus* para C++ junto con *ORBacus* para Java será necesario editar el fichero **config/sh.init** y realizar las modificaciones oportunas, siguiendo la ayuda que ofrecen los comentarios, sobre el directorio donde está instalado *JOB* y la máquina virtual Java que se utiliza.
- Ejecutar los mandatos devueltos por el *script* **runconfig**, (mostrados en el cuadro 1) bien tecleándolos tal cual en el *prompt* del sistema o con el ratón (*cut & paste*) seleccionando el listado desde la pantalla con el botón izquierdo y copiando al *prompt* pulsando el botón central (para ratón con tres botones) o con ambos botones a la vez (si tiene dos). Esto ejecutará el *script* **config.env** que contiene diferentes variables de entorno, eliminará cualquier configuración previa establecida y configurará (**./config.cache**) para nuestro sistema.
- Ejecutar **make** para iniciar el proceso de compilación.
- Ejecutar **make install** para instalar los elementos generados en sus directorios correspondientes.

Finalizada esta instalación, se dispondrá de los recursos necesarios para

Listado 1. Fichero Hola.idl

```
// Interfaz Hola.idl

interface Hola
{
void hola();
};
```

desarrollar aplicaciones distribuidas con *ORBacus* en C++. Se habrán instalado las utilidades **idl** y **jidl** para "mapear" a C++ y Java respectivamente, **hidl** y **ridl** para generar documentación en formato *HTML* y *RTF*; los servicios de Nombrado, Eventos y Propiedades; la librería que implementa al *ORB* y los ficheros de cabecera (*includes*) necesarios para utilizar dicha librería.

Los programas clientes harán uso de los objetos distribuidos como si de objetos locales se tratasen

Si durante la instalación se presentan problemas o se desea instalar *ORBacus* para otra plataforma o sistema operativo se pueden consultar los ficheros de instalación y ayuda del directorio **OB-3.1b1** y **JOB-3.1b1**. Si aún así tienes problemas puedes contactar con **support@ooc.com**.

INSTALACIÓN DE ORBACUS PARA JAVA

Para la distribución de Java los pasos necesarios para su instalación son:

- Descomprimir los fuentes con **tar xvfz JOB-3.1b1.tar.gz** y cambiar al directorio generado.
- Editar los ficheros **config/Make.rules** y **config/sh.init** y adecuarlos a la configuración de nuestro sistema. (Seguir las instrucciones que aparecen en los comentarios de los ficheros para realizar los cambios apropiados para tu instalación.) Será necesario especificar el directorio donde instalar los paquetes, la maquina virtual Java (por defecto *JDK*) y la utilidad **jidl.**
- Ejecutar **make** para iniciar el proceso de compilación.
- Ejecutar **make install** para instalar los paquetes compilados al directorio seleccionado en **config/Makerules.**

INSTALACIÓN DE GCC 2.8.X EN LINUX

La ultima versión del compilador de C *GNU*, la 2.8.1 puede obtenerse en cualquier servidor *ftp* que incluya aplicaciones bajo licencia *GNU*. En nuestro caso, se ha optado por el servidor *FTP* de *Red Iris*, **ftp.rediris.es**, que dispone del fichero **gcc-2.8.1.tar.gz** correspondiente a la versión en cuestión

La instalación del compilador no es un proceso complicado, aunque sí tedioso, y dependiendo de la máquina sobre la que se instala puede ser un proceso largo. De manera resumida, aunque en la mayoría de los casos suficiente, se indican los pasos que seguir para su correcta instalación.

- Descomprimir el fichero **gcc-2.8.1.tar.gz** mediante **tar xvfz gcc-2.8.1.**
- Acceder al directorio **gcc-2.8.1** generado en el proceso de descompresión.
- Eliminar ficheros innecesarios con **make distclean.**
- Configurar el compilador para nuestra plataforma y sistema operativo (**configure**).
- Generar el compilador de C, C++ y *Objective C* mediante **make**.
- Copiar el compilador al directorio **stage1**, utilizando **make stage1.**
- Obtener los distintos ficheros ejecutables correspondientes al compilador con **make CC="stage1/ xgcc –Bstage1/" CFLAGS="-g -O2".**
- Instalar el compilador mediante **make install CC="stage1/xgcc –Bstage1/" CFLAGS="-g –O2"**, que copiará **cc1**, **cpp** y **libgcc.a** al directorio **/usr/local/ lib/gcc-**

Listado 2. Definición de clase.

```
// Definición de la Clase
// Hola_impl.h

#include <Hola_skel.h>

class Hola_impl : public Hola_skel
{
public:
   virtual void hola();
};
```

Listado 3. Implementación del objeto.

```
// Implementación del objeto
// Hola_impl.cpp

#include <OB/CORBA.h>
#include <Hola_impl.h>

void Hola_impl::hola()
{
cout << "Hola Mundo!" << endl;
}
```

lib/i586/gcc-2.8.1 (**i586** es la máquina y **gcc-2.8.1** es la versión); copiará **c++** y **gcc** al directorio **/usr/local/bin**, **gcc.1** a **/usr/local /man/man1** y las páginas de información a **/usr/local /info**.

UN 'HOLA MUNDO!' DISTRIBUIDO Y MULTIPLATAFORMA

Para mostrar un primer ejemplo de aplicación distribuida multiplataforma, nada mejor que recurrir al 'superconocido' *Hola Mundo!*, que suele ser el primer programa que se realiza cuando se conoce un nuevo lenguaje o entorno de programación.

En este caso se desarrollará un programa servidor en C++ capaz de recibir peticiones de un cliente dirigidas a un objeto *CORBA*, de manera que cada vez que dicho objeto reciba una petición se mostrará el susodicho mensaje en pantalla **Hola Mundo!**. Por otra parte, también se desarrollará un programa cliente en Java que realizará peticiones al objeto cada vez que el usuario así lo decida.

Los pasos para desarrollar la aplicación, o cualquier aplicación *CORBA* son:

- Definición de las interfaces de todos los objetos cuyos métodos y atributos se deseen dejar accesibles a posibles clientes.
- Generación de las interfaces (*stubs* y *skeleton*).
- Implementación del programa servidor.
- Implementación del programa cliente.

DEFINICIÓN DE LA INTERFAZ IDL

Tras la fase de análisis[2] y diseño necesarias en el desarrollo de cualquier aplicación informática, nos vamos a encontrar en situación de identificar los módulos y los objetos que actuarán de interfaz o interconexión entre los programas cliente y servidor. Para todos aquellos objetos que se desee que sus métodos (también conocidos como funciones o servicios) queden accesibles a otras aplicaciones será necesario definir una interfaz en un lenguaje que permita describir la funcionalidad del objeto, y una fácil y no ambigua traducción a un lenguaje de implementación.

El lenguaje propuesto por la norma *CORBA* es *IDL* (*Inteface Definition Language*), que presenta una sintaxis parecida a la definición de interfaces en Java,

Listado 4. Programa principal del servidor.

```
// Programa principal
// Saludos.cpp

#include <OB/CORBA.h>
#include <Hola_impl.h>
#include <fstream.h>

int main ( int argc, char *argv[], char*[]) {

    CORBA_ORB_var orb = CORBA_ORB_init(argc, argv);
    CORBA_BOA_var boa = orb -> BOA_init(argc, argv);

    Hola_var p = new Hola_impl;

    CORBA_String_var s = orb -> object_to_string(p);
    const char* refFile = "Hola.ref";
    ofstream out(refFile);
    out << s << endl;
    out.close();

    boa -> impl_is_ready(CORBA_ImplementationDef::nil());
}
```

aunque con algunas diferencias. Actualmente están disponibles las especificaciones del mapeo de *IDL* hacia un buen número de lenguajes de implementación, orientados y no orientados a objetos. No es objetivo de este artículo presentar la sintaxis del lenguaje *IDL*, aunque el lector podría hacer uso de las referencias finales para conocer más sobre él.

En nuestro caso la aplicación servidor es muy simple y no es necesario ningún análisis previo, por lo que sólo tendremos que definir una interfaz. El Listado 1 muestra el contenido del fichero **hola.idl** que describe la interfaz **Hola** con una única operación definida *hola()*, que no toma ni devuelve valor alguno.

GENERACIÓN DE LAS INTERFACES. (STUBS Y SKELETONS)

El siguiente paso, consiste en un proceso de traducción, a veces conocido como compilación de la interfaz *IDL*, que genera las estructuras y clases necesarias (en definitiva código) para el enlace de la implementación del objeto *CORBA* correspondiente con el programa servidor (*skeleton*) y cliente (*stub*).

El *skeleton* se encargará de la recepción de solicitudes de métodos y parámetros y del envío de su correspondiente respuesta, todo, a través del *ORB*. El *stub* hará del sustituto (también conocido como *proxy*) del objeto en el cliente y se encargará de enviar solicitudes de métodos y recibir respuesta a través del *ORB*.

Para generar estos ficheros *ORBacus* dispone de una herramienta para cada lenguaje de implementación soportado; así, **idl** es la herramienta que permite el mapeo a C++ y **jidl** la que permite el mapeo a Java.

La orden **idl Hola.idl** generará los ficheros de cabecera e implementación del *skeleton* (**Hola_skel.h** y **Hola_skel.cpp**), y del *stub* (**Hola.h** y **Hola.cpp**).

Listado 5. Programa principal del cliente.

```
// Cliente java
// CSaludos.java

package PaqueteHola;

public class CCliente {

  public static void main(String args[]) {

    org.omg.CORBA.ORB orb =
        org.omg.CORBA.ORB.init(args, new java.util.Properties());

    String ref = null;

    try {
        String refFile = "Hola.ref";
        java.io.BufferedReader in =
          new java.io.BufferedReader(new FileReader(refFile));
        ref = in.readLine();
    } catch(java.io.IOException e) {
        System.err.println("Fichero no existe "+e.getMessage());
        System.exit(1);
    }

    org.omg.CORBA.Object obj = orb.string_to_object(ref);

    p.hola();
    }

}
```

La orden **jidl —package PaqueteHola Hola.idl** creará en el directorio **PaqueteHola** los ficheros **_HolaImplBase.java, HolaHolder.java, HolaHelper.java, Hola.java** y **StubForHola.java**.

IMPLEMENTACIÓN DEL PROGRAMA SERVIDOR

El desarrollo de la aplicación servidor tendrá que implementar y dejar accesibles los objetos *CORBA* para los que se ha definido una interfaz *IDL*. En el Listado 2, fichero **Hola_impl.h**, se muestra la definición de los objetos *CORBA* de nuestra aplicación, sólo uno en este caso. Además puede apreciarse que la clase *Hola_impl* deriva de la clase *Hola_skel* (generada automáticamente en el proceso de creación del *skeleton*), que a su vez deriva de la clase *CORBA*.

La implementación de los objetos para el ejemplo puede apreciarse en el Listado 3, fichero **Hola_impl.cpp**. En este caso, el método *hola* mostrará en pantalla el mensaje **Hola Mundo!**.

El Listado 4, fichero **Saludos.cpp** muestra la función *main* del servidor. Lo primero es obtener una referencia al *ORB* y al *BOA* (Adaptador Básico de Objetos) previa inicialización de algunos parámetros si se desea:

```
CORBA_ORB_var orb =
        CORBA_ORB_init(argc, argv);
CORBA_BOA_var boa = orb -> BOA_init(argc,
        argv);
```

Seguidamente se crearán instancias de los objetos que se van a a distribuir:

```
Hola_var p = new Hola_impl
```

Para permitir la accesibilidad hacia el objeto distribuido será necesario nombrar dicho objeto, asociándole una referencia única. Existen diversas técnicas para realizar dicha función de manera más o menos transparente. En este caso se ha optado por la solución más sencilla y también menos flexible.

```
CORBA_String_var s = orb ->
        object_to_string(p);
const char* refFile = "Hola.ref";
ofstream out(refFile);
out << s << endl;
out.close();
```

Con la función *object_to_string* se convierte la referencia a la implementación del objeto **Hola_impl** en una cadena de caracteres, permitiendo de este modo un fácil almacenamiento en un fichero. El principal inconveniente de este método reside en la necesidad por parte del cliente de acceder al fichero que almacena las identificaciones de los objetos distribuidos de nuestro sistema, que si bien en un entorno de pruebas no es un problema grave, sí resulta del todo inadecuado para entornos reales.

El siguiente paso consiste en dejar al servidor en espera de recibir solicitudes de clientes. Estas solicitudes serán tramitadas por el *BOA*.

```
boa -> impl_is_ready(CORBA_Implementation-
        Def::nil());
```

Finalmente se procederá al enlace (*link*) de todos los módulos que componen el programa servidor: **Saludos. cpp, Hola_skel.cpp, Hola_ impl.cpp** y **Hola.cpp** junto a las librerias del *ORB*.

IMPLEMENTACIÓN DEL PROGRAMA CLIENTE

Los programas clientes harán uso de los objetos distribuidos como si de objetos locales se tratase. La única diferencia residirá en cómo obtener la referencia a estos objetos. En el caso de un objeto local se obtiene al invocar al método constructor de la clase del objeto **p= new HolaImpl();** y en el caso de un objeto remoto a partir de solicitar al *ORB* la referencia a dicho objeto.

En nuestro caso el cliente se implementará en Java y su misión será realizar la invocación del método *hola* del objeto distribuido cuya referencia se encuentra almacenada en el fichero **Hola.ref.**

El Listado 5, fichero **Csaludos. java**, muestra la función *main* del cliente. Lo primero, al igual que en el caso del servidor consiste en obtener una referencia al *ORB* (el cliente no tiene asociado ningún Adaptador de Objetos).

```
org.omg.CORBA.ORB orb =
        org.omg.CORBA.ORB.init(args, new
java.util.Properties());
```

Posteriormente obtendrá del fichero que contiene las referencias de los objetos distribuidos[3] la cadena asociada al objeto remoto buscado.

```
String ref = null;
try {
        String refFile = "Hola.ref";
        java.io.BufferedReader in =
          new java.io.BufferedReader(new File-
        Reader(refFile));
        ref = in.readLine();
        } catch(java.io.IOException e) {
        System.err.println("Fichero no existe:
        "+e.getMessage());
        System.exit(1);
        }
```

Con la cadena que identifica al objeto remoto se puede obtener una referencia al mismo. La referencia obtenida es de la clase *CORBA*, por lo que es conveniente y a veces obligado la adecuación (*narrow*) al tipo de la clase en cuestión.

```
        org.omg.CORBA.Object obj =
        orb.string_to_object(ref);
Hola p= HolaHelper.narrow(obj);
```

Finalmente, la operación se invoca a la función *hola*, provocando que se muestre en la pantalla correspondiente del programa servidor el mensaje **Hola Mundo!.**

```
p.hola();
```

Para compilar la aplicación cliente, utilizando **javac**, será necesario compilar los fuentes generados en el **PaqueteHola** y el programa principal del cliente, ejecutando:

```
javac *.java PaqueteHola/*.java
```

Notas:

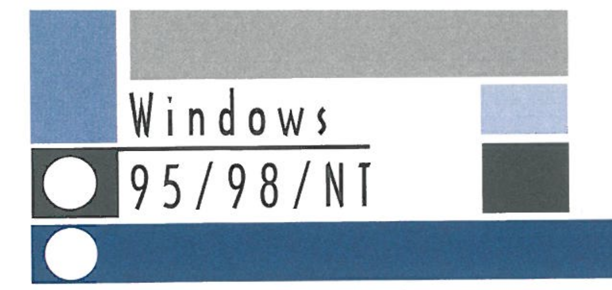
[1] Este programa cliente podrá ser iniciado, incluso simultáneamente, en tantos *hosts* como se deseen, incluso en el *host* donde se encuentra el servidor.

[2] Si se siguen las recomendaciones de la *OMG* se habrá utilizado el Lenguaje de Modelado Unificado (*UML*) para esta fase.

[3] Lo más adecuado sería hacer uso del servicio de nombrado para obtener una referencia al objeto remoto.

BIBLIOGRAFÍA

- Robert Orfali, Dan Harkey, Jery Edwards. "The Essential Distributed Objects Survival Guide". John Wiley & Sons. 1996.
- Alan Pope. "The CORBA Reference Guide". Addison-Wesley. 1997
- Manual de Usuario de ORBacus 3.



Windows Scripting Host (I)

Adolfo Aladro (aaladro@arrakis.es)

Prácticamente todos aquellos usuarios que hayan trabajado con MS-DOS conocen los archivos BAT. Se trata de archivos de proceso por lotes que se han venido utilizando para realizar tareas sencillas de mantenimiento del sistema (copiar, borrar, mover archivos o directorios, etc.) o pequeños programas.

Aunque los archivos *BAT* han sido y son todavía bastante utilizados, la verdad es que presentan muchos inconvenientes. El primero y quizá más limitador de todos nace de su propia simpleza. Para realizar archivos *BAT* contamos tan sólo con unas pocas estructuras de control y algunos comandos del sistema.

Con el paso del tiempo, los archivos *BAT* fueron heredados por Windows 95 y por lo tanto continuaron siendo utilizados. Sin embargo al calor de la irrupción de Internet los sistemas operativos han empezado a hacer suyas características que hasta entonces eran exclusivas de los navegadores.

WSH (*Windows Scripting Host*) es una tecnología que ha nacido dentro de esa tendencia y en resumen consiste en un intérprete de comandos nativo que es capaz de interpretar lenguajes de *script* que hasta ahora eran de uso exclusivo dentro de las tecnologías Internet: *JavaScript* y *VBScript*.

Una de las principales ventajas que presenta *WSH* es el hecho de que los lenguajes de *script* son muy conocidos e indudablemente su potencia con respecto a los viejos archivos *BAT*, al tratarse de lenguajes de alto nivel. A partir de ahora podremos crear programas en *JavaScript* o *VBScript* y ejecutarlos de forma nativa en los sistemas de Microsoft.

WSH viene incorporado en los nuevos sistemas operativos Windows 98 y Windows NT 5 y también es posible añadirlo a los sistemas Windows 95 y Windows NT 4 instalando un fichero que

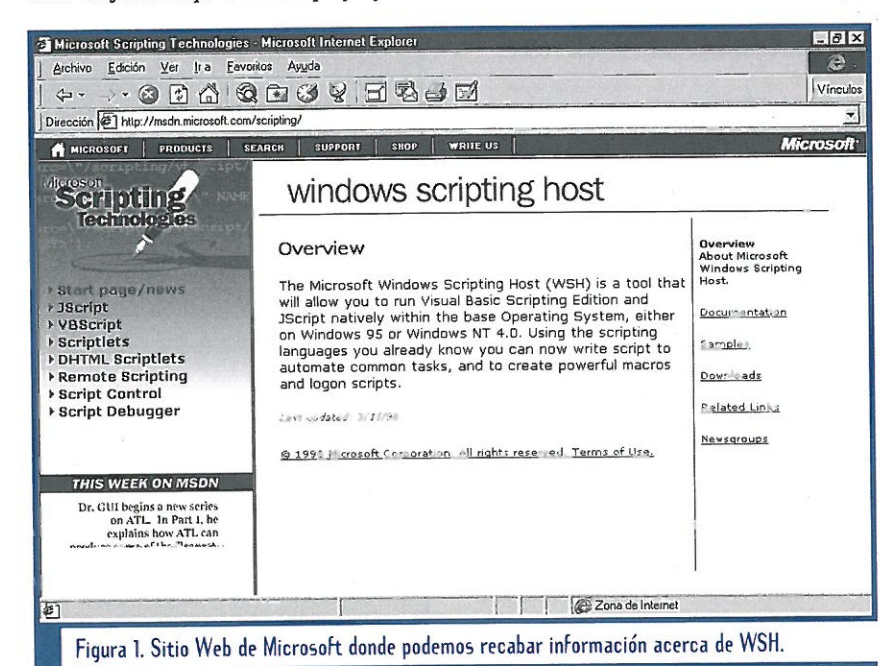


Figura 1. Sitio Web de Microsoft donde podemos recabar información acerca de WSH.

puede descargarse gratuitamente del *web* de Microsoft **http://www.microsoft.com/scripting**.

En cualquiera de los dos casos, una vez que tenemos el sistema instalado si pulsamos con el ratón en un archivo con extensión *JS* (*JavaScript*) o *VBS* (*VBScript*) podremos acceder a algunas propiedades de ejecución del *script* en cuestión (Ver **figura 2**).

Nuestros programas *script* se podrán ejecutar de dos formas: en modo línea de comando (en este caso será el ejecutable **Cscript.exe** el responsable de ejecutarlos) o en modo normal, tal y como ocurre cuando se hace doble clic sobre su icono dentro del entorno Windows (En este caso el responsable de ejecutarlos será el programa **WScript.exe**).

WSH no sólo ofrece la posibilidad de utilizar lenguajes como *JavaScript* y *VBScript* sino que además pone a disposición del programador una serie de objetos nuevos a través de los cuales podremos conocer características de nuestro sistema. A continuación introduciremos algunos de esos objetos y veremos sus propiedades y métodos. En cualquier caso se recomienda al lector que obtenga la documentación completa de todos ellos en el *web* de Microsoft.

EL OBJETO WSCRIPT

Este objeto puede ser utilizado en cualquier momento dentro de un *script* y nos proporciona información acerca del propio *script* que se está ejecutando. Veamos a continuación sus propiedades.

- *Arguments*
  Objeto que contiene la colección de parámetros en línea que ha recibido el *script*.

- *FullName*
  Contiene la ruta completa del programa que está ejecutando el *script*, es decir, la ruta del programa **wscript.exe** (si se está ejecutando en el entorno *Windows*) o bien la del programa **cscript.exe** (si se está ejecutando en línea de comandos).

- *Name*
  Contiene el nombre del objeto *Wscript*. Normalmente será **Windows Scripting Host.**

- *Path*
  Contiene la ruta completa del programa que está ejecutando el *script*. Es exactamente igual que *FullName* pero devuelve sólo la ruta sin el nombre.

- *ScriptFullName*
  Contiene la ruta completa del propio *script*.

- *ScriptName*
  Igual que el anterior pero contiene sólo el nombre.

- *Version*
  Contiene el número de versión de *WSH*.

El Listado 1 contiene un sencillo ejemplo de un programa script que muestra una ventana de alerta con los valores de las propiedades del objeto WScript. La figura 3 muestra el resultado de ejecutar este programa script.

En el programa anterior ya hemos utilizado un método del objeto *WScript*,

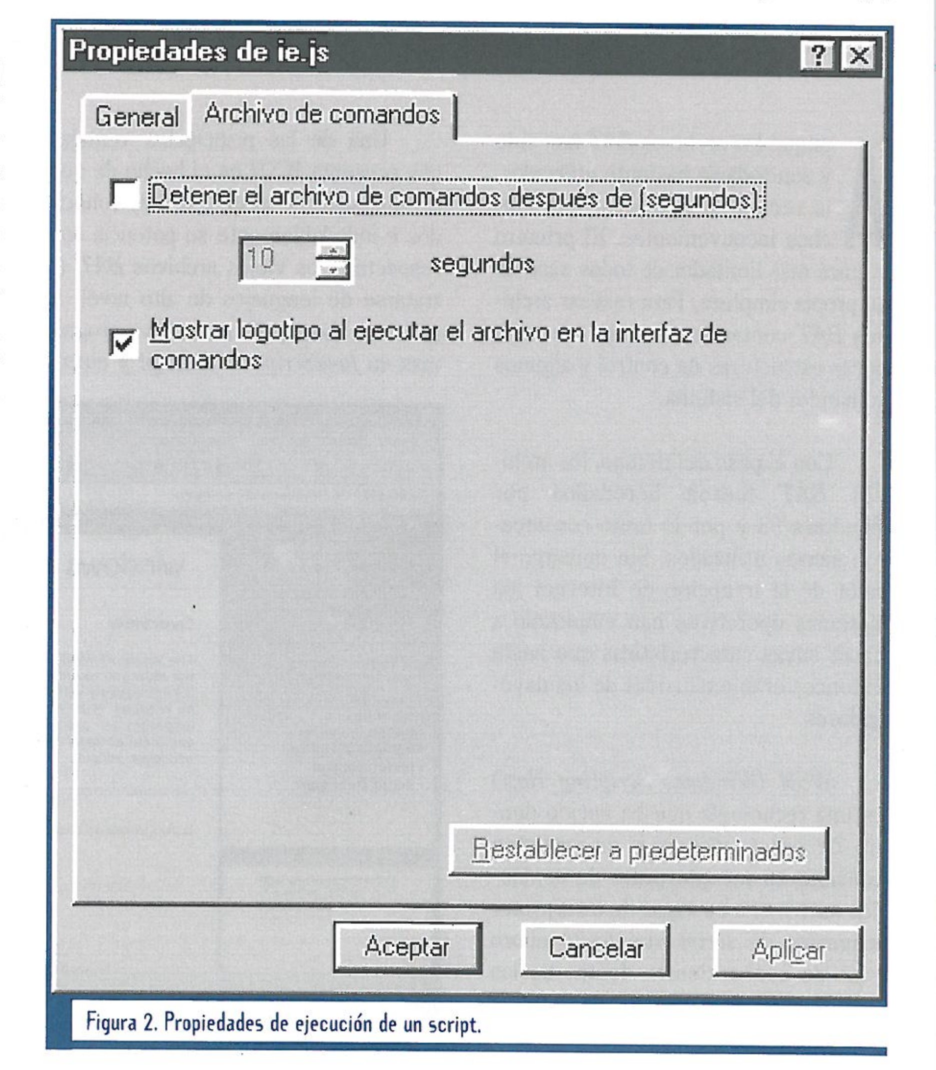


Figura 2. Propiedades de ejecución de un script.

WScript.Echo. Veamos a continuación todos los métodos del citado objeto.

- *CreateObject*, *DisconnectObject* y *GetObject*
Estos tres métodos se utilizan para manejar objetos *COM*. De esta forma se pueden automatizar todo tipo de aplicaciones. Así por ejemplo:

```
var objXL =
    WScript.CreateObject("Excel.Application
    ");
```

Hace que la variable **objXL** contenga un objeto del tipo aplicación Excel gracias al cual podemos tener acceso prácticamente a casi todas las propiedades de dicha aplicación.

- *Echo*
Este método se utiliza para lanzar ventanas de alerta con mensajes.

- *Quit*
Este método hace que se detenga la ejecución del programa actual.

EL OBJETO WSHSHELL

Este objeto se utiliza para conocer información sobre del sistema. Si bien el objeto anterior reside en la propia aplicación *WSH*, el objeto *WshShell* necesita tener una instancia antes de poder hacer referencia a él, ya que se trata de un control *OCX*.

- *Enviroment*
Contiene un objeto de tipo *WshEvironment* que es una colección de cadenas que contienen todas las variables de entorno del sistema. En el **Listado 2** podemos ver el código de un programa que muestra todas las variables de entorno.

Listado 1. Programa Script que muestra las propiedades del objeto WScript en una ventana de alerta.

```
var tmp = "WScript.FullName:\t\t" + WScript.FullName + '\n';
tmp += "WScript.Name:\t\t" + WScript.Name + '\n';
tmp += "WScript.Path:\t\t" + WScript.Path + '\n';
tmp += "WScript.ScriptFullName:\t" + Wscript.ScriptFullName + '\n';
tmp += "WScript.ScriptName:\t" + WScript.ScriptName + '\n';
tmp += "WScript.Version:\t\t" + WScript.Version;
WScript.echo(tmp);
```

Listado 2. Programa script que muestra la variable de entorno del sistema.

```
Set WshShell = Wscript.CreateObject("Wscript.Shell")
For Each strVarName In WshShell.Environment("PROCESS")
MsgBox strVarName
Next
```

- *SpecialFolder*
Tiene un objeto *WshSpecialFolders* gracias al cual podemos acceder a las carpetas especiales del sistema.

Veamos a continuación todos los métodos del objeto *WshShell*.

- *CreateShortCut*
Este método sirve para crear un objeto de tipo *WshShortcut* (Un acceso directo). Si el nombre del objeto termina con "**.URL**" entonces se crea un objeto de tipo *WshURLShortcut* (Dirección URL). En el **Listado 3** podemos ver el código de un programa de ejemplo que crea un acceso directo al propio programa de *script*.

- *ExpandEnviromentsStrings*
Este método permite ver el contenido de las variables de entorno del proceso actual. Así, el **Listado 4** muestra cómo podríamos obtener por ejemplo el valor de la variable *TMP*.

- *Popup*
Este método permite mostrar un mensaje al usuario y esperar su respuesta durante un tiempo determinado.

- *RegDelete*, *RegRead* y *RegWrite*
Estos tres métodos sirven para acceder al registro de configuraciones del sistema. Los dos primeros toman el nombre de una

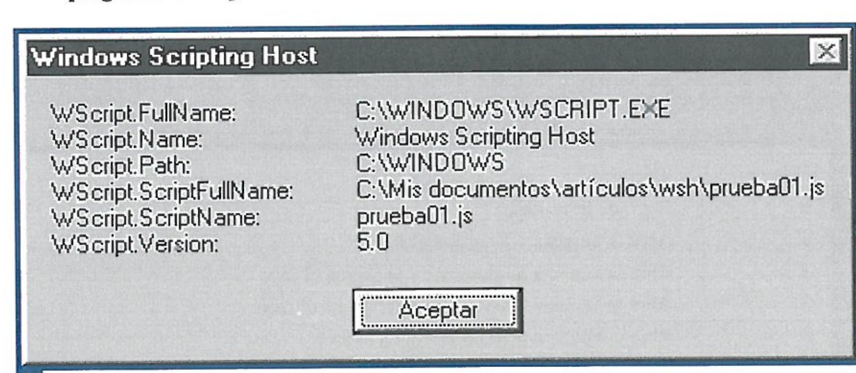


Figura 3. Resultado de la ejecución del programa que figura en el Listado 1.

entrada del registro y lo utilizan para borrar o leer su contenido respectivamente. La **Tabla 1** muestra algunas abreviaturas que se pueden utilizar para acceder a ciertas claves del registro.

El método *RegWrite* se utiliza para escribir entradas en el registro. Hay que especificar la ruta, el valor y el tipo de los datos que se van a escribir (*REG_SZ, REG_EXPAND_SZ, REG_DWORD* o *REG_BINARY*).

- *Run*
  Este método permite lanzar otros procesos. El **Listado 5** muestra un ejemplo de un programa que abre el *Notepad*. Existe un segundo parámetro opcional que indica el tipo de ventana en la que se ejecutará el programa. La **Tabla 2** muestra algunos de los distintos valores que puede tomar este parámetro. Finalmente hay un tercer parámetro opcional de tipo *booleano* que permite elegir entre esperar a que el proceso termine o devolver inmediatamente el control al programa *script*.

Listado 3. Programa script que crea un acceso directo.

```
var WshShell = Wscript.CreateObject("WScript.Shell");
var oShellLink = WshShell.CreateShortcut("Current Script.lnk");
oShellLink.TargetPath = WScript.ScriptFullName;
oShellLink.Save();
```

Listado 4. Acceso a los valores de las variables de entorno.

```
var WshShell = WScript.CreateObject("WScript.Shell");
WScript.Echo(WshShell.ExpandEnvironmentStrings("%TMP%"));
```

Listado 5. Inicio de otros procesos a través de un de script.

```
var WshShell = WScript.CreateObject("Wscript.Shell");
WshShell.Run("notepad.exe");
```

MANEJANDO DIRECTORIOS Y ARCHIVOS

Hasta aquí hemos visto brevemente las propiedades y métodos de algunos de los objetos nuevos que proporciona *WSH*. Sin embargo las verdaderas de WSH no procede de estos nuevos objetos, sino de la potencia de los propios lenguajes de *script*. A continuación vamos echar un vistazo a una de las tareas que desempeñaremos con más frecuencia utilizando programas de *script*: el manejo de directorios y archivos.

Los lenguajes *JavaScript* y *VBScript* ponen a nuestra disposición numerosos objetos gracias a los que podremos acceder al sistema de archivos del sistema y hacer cosas básicas como ver el contenido de directorios; copiar, mover o borrar directorios; copiar, mover o borrar archivos; etc. La **Tabla 3** muestra un resumen de los mismos. Ver todas las propiedades y los métodos de estos objetos excede con mucho la extensión de este artículo así que procuraremos adentrarnos en el tema a través de un par de ejemplos sencillos.

El Listado 6 muestra un programa *script* que visualiza todos los directorios que "cuelgan" del directorio actual.

En primer lugar vamos a crear una instancia de un objeto del tipo

Tabla 1. Abreviaturas que se pueden emplear en el acceso a entradas del registro de configuraciones del sistema.

Abreviatura	Clave
HKCU	HKEY_CURRENT_USER
HKLM	HKEY_LOCAL_MACHINE
HKCR	HKEY_CLASSES_ROOT
	HKEY_USERS
	HKEY_CURRENT_CONFIG

Tabla 2. Los distintos tipos de ventana en la que se puede ejecutar una aplicación lanzada desde un programa de script.

Valor	Tipo
0	Esconde la ventana
1	Abre la ventana y le asigna el foco
2	Abre la ventana minimizada y le asigna el foco
3	Abre la ventana maximizada y le asigna el foco
4	Abre la ventana pero no le asigna el foco
5	Abre la ventana minimizada y no le asigna el foco

FileSystemObject el cual nos dará acceso a todas las propiedades del sistema de archivos del sistema.

```
fso = new
    ActiveXObject("Scripting.FileSystemObje
    ct");
```

Posteriormente gracias al método *GetFolder* creamos una instancia de un objeto de tipo *Folder*.

```
f = fso.GetFolder("c:");
```

Como el parámetro que le pasamos es "c:" el directorio que se tomará será aquel donde se encuentre el propio programa script, es decir, el directorio por defecto. Finalmente accedemos a la propiedad SubFolders para ver todos los directorios que "cuelgan" del directorio que anteriormente hemos seleccionado y los mostramos.

El Listado 7 muestra otro ejemplo más. En este caso se trata de un programa *script* que visualiza el contenido de un archivo de texto. Como podemos observar volvemos a crear una instancia de un objeto del tipo *FileSystemObject* pata tener acceso a todas las propiedades del sistema de archivos del sistema. A partir de este objeto abrimos un archivo y visualizamos su contenido.

Como hemos visto a lo largo de este artículo introductorio *WSH* (*Windows Scripting Host*) resulta una forma excelente de aprovechar las características de lenguajes avanzados, cuyo uso se encuentra muy extendido en el ámbito de Internet, para realizar todo tipo de tareas de mantenimiento del sistema. El avance que se produce con respecto a los antiguos archivos *BAT* es más que considerable y por primera vez el sistema *Windows* tiene una interfaz de comandos realmente útil, potente e integrada. En el próximo artículo trataremos de acercarnos a ejemplos más complejos gracias a los cuales podremos formarnos una mejor idea global sobre las posibilidades de esta tecnología.

Tabla 3. Objetos que proporcionan los lenguajes de script para manejar ficheros y directorios.

Objeto	Descripción
Drive	Proporciona acceso a las propiedades de un dispositivo (un disco, un dispositivo de red, etc.).
Drives	Proporciona una colección con todos los dispositivos del sistema.
File	Proporciona acceso a todas las propiedades de un archivo.
Files	Proporciona una colección de todos los objetos de tipo File dentro de un directorio.
FileSystem Object	Proporciona acceso al sistema de ficheros del ordenador.
Folder	Proporciona acceso a todas las propiedades de un directorio o carpeta.
Folders	Proporciona la colección de objetos de tipo Folder que se encuentra dentro de un objeto de tipo Folder.

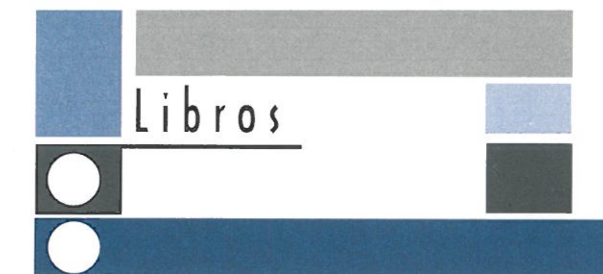
Listado 6. Programa script que muestra todos los directorios que "cuelgan" del directorio actual.

```
var fso, f, fc;
var tmp ="";
fso = new ActiveXObject("Scripting.FileSystemObject");
f = fso.GetFolder("c:");
fc = new Enumerator(f.SubFolders);
for (; !fc.atEnd(); fc.moveNext()) {
  tmp += fc.item();
  tmp += "\n";
}
WScript.Echo(tmp);
```

Listado 7. Programa script que muestra el contenido de un archivo de texto.

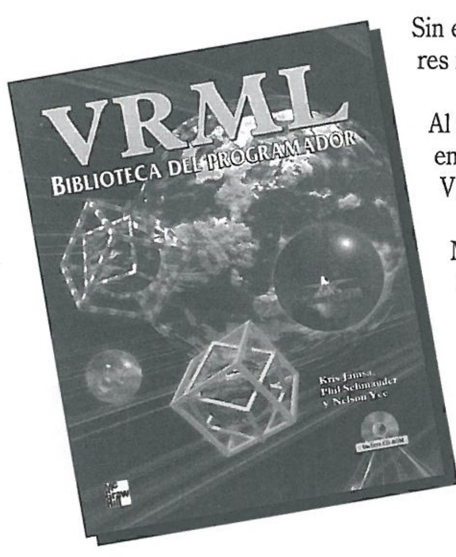
```
var fso, f;
var ForReading = 1, ForWriting = 2;
var result = "";
var archivo = "ejemplo.txt"

fso = new ActiveXObject("Scripting.FileSystemObject");
f = fso.OpenTextFile(archivo, ForReading);
while(!f.AtEndOfStream) {
  linea = f.ReadLine();
  result += linea + "\n";
}
f.close();
WScript.Echo(result);
```



VRML BIBLIOTECA DEL PROGRAMADOR

En tiempos remotos, pocos o casi ningún navegador soportaba VRML y tristemente, pocos sitios en Internet mostraban algún objeto tridimensional.



Sin embargo, en la actualidad ya están disponibles auxiliares de VRML para los populares navegadores Netscape Navigator y Microsoft Internet Explorer.

Al comenzar con mundos simples basados en VRML 1, esta obra le enseña a pensar en términos tridimensionales y le proporciona una sólida base para programar con VRML, consiguiendo así el objetivo deseado.

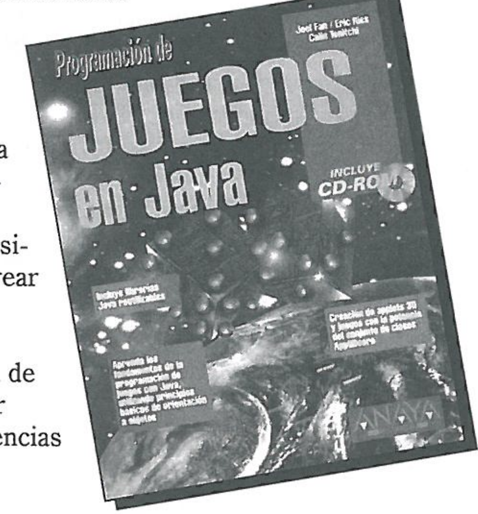
Más adelante utilizando VRML 2, podrá introducir sonido, fondos, sensores digitales e incluso objetos animados en sus mundos virtuales.

Este libro le proporcionará miles de líneas de código fuente y todo lo que necesita para crear mundos virtuales.

Editorial: McGraw-Hill	Autor: Kris Jamsa, Phil Schmauder y Nelson Yee
Nº páginas: 539	Idioma: Español
Nivel: Intermedio-avanzado	Precio: 6.000 Ptas.(I.V.A. inc.)

PROGRAMACIÓN DE JUEGOS EN JAVA

Este libro le revela como crean juegos Java interactivos para la Web los maestros de la codificación. Con él conseguirá estar en la vanguardia de laya denominada revolución de los juegos.



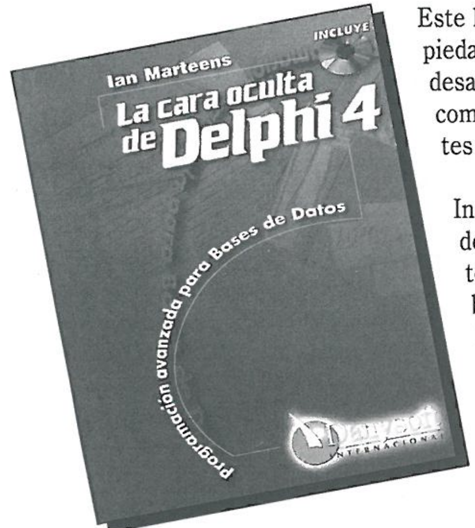
Combinará los principios del diseño de juegos orientados a objetos con la capacidad y potencial de lenguaje de programación Java para crear entretenimientos dinámicos en línea. Además, desvela el misterio del sonido, animación, color, gestión de red, gráficos avanzados y juegos multijugador, lo que le permite que sea el único libro que necesitará para programar juegos interactivos con los últimos adelantos o crear efectos para la red Internet similares a los de los juegos utilizando la última versión de Java.

La sección Galería de juegos le proporciona una visión exclusiva de algunos de los programas Java existentes más recientes. Puede añadir potencia a su motor de gráficos mediante la utilización de clases, herencias y enlace dinámico de métodos.

Editorial: Anaya Multimedia	Autor: Joel Fan, Eric Ries y Calin Tenitchi
Nº páginas: 840	Idioma: Español
Nivel: Intermedio-avanzado	Precio: 5.995 Ptas.(I.V.A. inc.)

LA CARA OCULTA DE DELPHI 4

Este libro le guiará en el aprendizaje de la programación para bases de datos con Delphi: desde la Programación orientada a Objetos, el Modelo de Componentes y las técnicas para grupo de desarrollo hasta los últimos avances del modelo COM/DCOM y la tecnología Midas para aplicaciones en múltiples niveles.



Este libro va más allá de la simple y llana exposición de métodos, eventos y propiedades, ya que le propone unas técnicas de integración y una metodología de desarrollo válidas tanto para aplicaciones monosaurios o en redes punto a punto, como para complejas aplicaciones cliente/servidor con transacciones concurrentes y actualizaciones en caché.

Incluye capítulos sobre Oracle 8 y sus extensiones de objetos, MS SQL Server, desarrollo para Internet, implementación por el BDE de la comunicación cliente/servidor, extensiones de InstallShield Express2 y adem;as las nuevas posibilidades de Midas incorporadas en la versión 4 de Delphi.
El libro va acompañado de un CD-ROM

Editorial: Danysoft Internacional	Autor: Ian Marteens
Nº páginas: 912	Idioma: Español
Nivel: Avanzado	Precio: 7.900 Ptas.(I.V.A. inc.)

A FONDO MICROSOFT VISUAL INTERDEV

Si quiere construir sitios web de alto rendimiento o crear intranets, el sistema de desarrollo web de Microsoft Visual InterDev es la respuesta.

Con Visual InterDev, obtendrá sofisticadas opciones de conectividad de bases de datos, capacidad de gestión de sitios, y una estrecha integración con otras herramientas de Microsoft como el sistema de programación Visual Basic, el sistema de desarrollo de Visual C++ y el software de desarrollo de Visual J++.

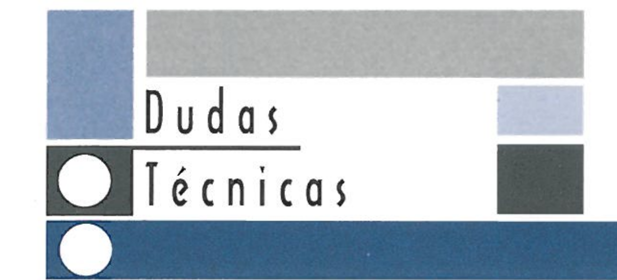
Este libro le proporciona una perspectiva desde dentro de esta poderosa herramienta de desarrollo. Además le muestra cómo poner en funcionamiento estas potentes características:

Controles y scripts ActiveX en el cliente y en el servidor, Active Server Pages, integración con bases de datos ODBC, editor HTML WYSIWYG, amplio soporte de lenguajes, ricas herramientas multimedia, etc.

El CD-ROM que acompaña al libro contiene una versión de evaluación de Microsoft Visual InterDev, Microsoft Internet Explorer 4.01(incluyendo SDK) así como numerosos ficheros de ejemplo.



Editorial: McGraw-Hill	Autor: Ken Miller, Ken Spencer, Eric Vincent y N.D.Evans
Nº páginas: 516	Idioma: Español
Nivel: Avanzado	Precio: 7.500 Ptas.(I.V.A. inc.)



Correo del lector

En esta sección, los lectores de SÓLO PROGRAMADORES tienen la oportunidad de hallar respuesta a los problemas que puedan tener en cualquier tema relacionado con la programación en cualquier entorno.

Pregunta

La presente me da la oportunidad de felicitarles por la gran calidad de su revista ya que su contenido es de gran ayuda para mantenerse actualizado en este cambiante mundo de la informática.

Además me agradaría que me brindaran su ayuda ya que cuento con una pequeña aplicación en DELPHI y quiero que al inicializarla se aloje en la barra de tareas de WINDOWS 95 dentro del recuadro donde se ubica el reloj y la configuración del idioma. Me sería de gran ayuda su respuesta.

Respuesta

Para inicializar una aplicación en la barra de tareas en Delphi hacen falta básicamente dos cosas: la primera es fijar a "False" la propiedad "ShowMainForm" del objeto "Application". Con esto forzamos que no se muestre ninguna ventana al inicio de la aplicación.

La segunda es llamar a la función de la API de Windows: Shell_NotifyIcon. Para llamar a esta función, previamente hay que rellenar un registro del tipo "TNotifyIconData", definid en la "unit": "Shellapi" y cuyas variables son las siguientes:

- cbSize: Tamaño del registro
- Wnd: Handle de la ventana de la aplicación
- uID: Identificador
- hIcon: Dibujo del Icono (es un Handle)
- szTip: "Hint" del Icono
- uCallbackMessage: Mensaje que enviará el icono al recibir un evento
- uFlags : Cuáles de las 3 ultimas variables anteriores están rellenadas (NIF_ICON, NIF_TIP, NIF_MESSAGE)

Luego se hace una llamada a la función "Shell_NotifyIcon". Para añadir el icono a la barra de tareas, la llamada será:

```
Shell_NotifyIcon (NIM_ADD, @DatosIcono);
```

Para borrar el icono, la llamada será:

```
Shell_NotifyIcon (NIM_DELETE, @DatosIcono);
```

La mejor manera de entenderlo es con un ejemplo. Creamos un formulario que llamamos "TFormIcono" en una unit "TUnitIcono". LLamamos al proyecto: "AppIcono". En el formulario ponemos una etiqueta (Label1) y un botón (Ocultar).

Añadimos un "PopupMenu" con dos opciones de menú: una que se llame "Cerrar" y otra que se llame "Mostrar". El código completo del formulario de ejemplo se muestra a continuación (está comentado completamente para que sea autoexplicativo):

```
unit UnitIcono;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
  StdCtrls, Buttons, Shellapi, Menus;

const
  { Identificador del Icono y Mensaje del Icono }
  IconoELastra = 100;
  UM_ICO_ELASTRA = WM_USER + 100;

type
  TFormIcono = class(TForm)
  { Etiqueta con un título }
  Label1: TLabel;

  { PopupMenu y sus dos opciones de menú }
  PopupMenu1: TPopupMenu;
  Cerrar: TMenuItem;
  Mostrar: TMenuItem;
```

```
  { Botón para ocultar }
  Ocultar: TBitBtn;

  procedure FormCreate(Sender: TObject);
  procedure FormClose(Sender: TObject; var
     Action: TCloseAction);
  procedure CerrarClick(Sender: TObject);
  procedure MostrarClick(Sender: TObject);
  procedure OcultarClick(Sender: TObject);
private
  { Datos del Icono de la barra de tareas }
  DatosIcono: TNotifyIconData;

  { Manejador del mensaje enviado por el Icono
     de la barra de tareas }
   procedure MostrarIcoELastra (var Mensaje:
     TMessage); Message
UM_ICO_ELASTRA;
   end;

var
  FormIcono: TFormIcono;

implementation

{$R *.DFM}

procedure TFormIcono.FormCreate(Sender:
        TObject);
begin
  { No mostramos la ventana principal al cargar
     la aplicación }
  Application.ShowMainForm := False;

  { Rellenamos los datos del Icono }
  with DatosIcono do begin
     { Asignamos los parámetros obligatorios }
     cbSize := SizeOf(DatosIcono);
     Wnd   := Self.Handle;
     uID   := IconoELastra;

     { Asignamos el dibujo del Icono }
     hIcon := Application.Icon.Handle;

     { Asignamos el "Hint" del Icono }
     szTip := 'Aplicación de Enrique de la
     Lastra';

     { Asignamos el mensaje que enviará el
     icono }
     uCallbackMessage := UM_ICO_ELAS-
     TRA;

     { Rellenamos TODOS los parámetros:
     ¡Ojo! porque si realizamos un
      "AND" en lugar de un "OR", no obtendría-
     mos el mismo resultado }
     uFlags := NIF_ICON or NIF_TIP or
     NIF_MESSAGE;
   end;

  { Añadimos el Icono a la barra de tareas }
  Shell_NotifyIcon (NIM_ADD, @DatosIcono);
end;

procedure TFormIcono.FormClose(Sender: TOb-
        ject; var Action:
TCloseAction);
begin
  { Borramos el Icono de la barra de tareas }
Shell_NotifyIcon (NIM_DELETE, @DatosIcono);
end;

procedure TFormIcono.MostrarIcoELastra (var
        Mensaje: TMessage);
var
  Punto: TPoint;
begin
  { Procesamos el mensaje que ha enviado el
     Icono de la barra de tareas }
  with Mensaje do begin
  { Si el usuario pulsa el botón izquierdo, mos-
     tramos la ventana de la aplicación }
  if LParam = WM_LBUTTONDOWN then
     begin
     Application.ShowMainForm := True;
     Show;
  end
  { Si el usuario pulsa el botón derecho, mostra-
     mos un menú contextual }

  else if LParam = WM_RBUTTONDOWN
     then begin
     GetCursorPos(Punto);
     PopupMenu1.Popup (Punto.X,
      Screen.Height)
  end;
 end;
end;

procedure TFormIcono.CerrarClick(Sender:
        TObject);
begin
 { Cerramos la aplicación }
 Close;
end;

procedure TFormIcono.MostrarClick(Sender:
        TObject);
begin
 { Mostramos la ventana principal }
 Show;
end;

procedure TFormIcono.OcultarClick(Sender:
        TObject);
begin
 { Ocultamos la ventana principal }
 Hide;
end;

end.
```

Pregunta

Desde sus comienzos, soy un asiduo lector de su fenomenal publicación, muchas gracias por ayudarnos a continuar en esta profesión.

Actualmente estoy aprendiendo a programar en DELPHI 3.0, ¿me podríais aconsejar algún buen libro, que describa los componentes y sobre todo que tenga muchos ejemplos?

Tengo una segunda duda y mi pregunta es: ¿cómo se pueden cambiar los mensajes mostrados en inglés por el componente Query, cuando muestra el cuadro de diálogo con la progresión de la impresión?

Y por ultimo, quisiera saber qué "comandos" "tag" he de utilizar para crear archivos de ayuda con WORD, y utilizar el compilador de ficheros de ayuda HCW.

Respuesta

Un buen libro de programación en Delphi es el de Francisco Chartre, se trata de un manual muy didáctico y lo más importante, es de un autor español.

Respondiendo a tu segunda pregunta te diré que los mensajes del

TQuery, como todos los de los componentes de Delphi vienen en inglés aunque hay librerías traducidas al español.

En cuanto a los archivos de ayuda creados con WORD basta guardarlos en formato RTF. La última versión de Word (Word 97) genera un RTF que alintentar compilarlo con el HCW éste informa que es erróneo. Utiliza la versión 6.0 de Word.

Pregunta

¿Existe algún compilador y alguna máquina virtual de java para MS- DOS?.

En caso de ser una pregunta afirmativa, ¿podrían incluirlo en su material magnético junto a la revista?. El sentido de programar en java bajo MS-DOS es el de aprovechar aquel anticuado ordenador que apenas utilizamos pues no sabemos en qué, y de esta manera será una buena excusa para poder aprender un lenguaje en auge.

Muchas gracias

Respuesta

Para que una máquina virtual en MS-DOS sea al menos compatible la especificación 1.0.2 de Java debe ofrecer un entorno gráfico, el AWT, y como todos recordamos, MS-DOS no se caracterizaba por ello, por lo que su implementación a priori parece una tarea complicada, ya que no se define bajo un standard a nivel de sistema operativo. Por tanto, inicialmente no tiene mucho sentido la existencia de máquinas virtuales para ese entorno.

Si se quiere utilizar Java con MS-DOS para aprovechar esa vieja máquina que todos tenemos por ahí, esa máquina debe ser como mínimo un 486, ya que las máquinas virtuales Java hacen un uso "intensivo" del coprocesador, e incluso con ello, un 486 ofrecería rendimiento pésimo.

Una buena alternativa es instalar una "versión reducida" o antigua de Linux, que si ofrece un rendimiento razonable con un 486 utilizando las JDK disponibles.

Como alternativa en el mundo Windows se puede utilizar el compilador DJGPP para MS-DOS e intentar compilar GUAVAC, una implementación de libre distribución disponible en *http://cs.berkeley.edu/~enberg/guavac.html*. En la compilación por regla general se requieren "ddl's" que solo funcionan bajo Windows, por lo que esta alternativo no parece factible.

Existe una implementación para Windows 3.1, del grupo AlphaWorks de IBM *(http://alphaworks.ibm.com)* que se denomina ADK. Es una implementación al completo de la versión 1.0.2, operativa al 100%, que requiere muy poca memoria y procesador y lo mejor de todo, totalmente gratuita.

También se puede intentar compilar con DJGPP una máquina virtual, Kaffe, disponible en *http://www.kaffe.org* y no esperar muchos resultados.

Pregunta

He leido con gran interés el artículo de Francisco Javier Toledo publicado en el número 50 de Sólo Programadores en el que comparaba las características principales de dos sistemas operativos de actualidad: Windows NT y Unix.

A pesar de la información que aporta el artículo, no tengo claro cual de los dos sería más adecuado para mi trabajo, consistente en dar clases de Informática.

Administro una sencilla red en un instituto público que da servicio a 12 puestos (486) con Windows 3.11 para trabajo en grupo. En el servidor (un Pentium MMX) tenemos Windows 95, y lo usamos como servidor de impresión, servidor de aplicaciones y ficheros, servidor proxy (Wingate) y servidor web de la Intranet (Microsoft Personal Web Server). Trabajamos principalmente con programas ofimáticos (Lotus Smart Suite), navegación por Internet y practicas de HTML.

Windows 95 me plantea problemas de seguridad (no puedo restringir los accesos de forma adecuada a cada área del servidor) y de fiabilidad (es necesario reiniciar el equipo a menudo), por lo que he pensado cambiarlo por Windows NT o Linux. Sin embargo no sé si al utilizar estos sistemas podré seguir instalando aplicaciones en el servidor (el Smart Suite) para ejecutarlas desde los puestos, y esto es muy importante para mí, ya que los puestos están bastante escasos de capacidad de disco y no podemos plantearnos a corto plazo una gran inversión.

Respuesta

En principio si quieres seguir utilizando el software que tienes en este mismo momento deberías optar por una solución basada en Windows NT con NTFS como sistema de partición y si fuese posible Winsdows 95 o Workstation como clientes. Ni Wingate ni Lotus Smart Suite deberían darte ningún problema (obviamente depende de las versiones que tengas de los mismos). Una vez instalado NT podrías sustituir Microsoft Personal web Server por Internet Information Server que viene incluido con este sistema operativo y tiene un carácter mas profesional. Tus problemas de seguridad quedarían solucionados gracias al sistema de ficheros NTFS. Para no tener que reiniciar el servidor lo mejor es tener instalados los services paks y es muy importante tener un hardware de calidad (evitemos el peligro de los clónicos).